Cycle time analysis of photolithography systems in a semiconductor manufacturing plant

Spierings, J.T.

Award date:
2013

Cycle time analysis of photolithography systems in a semiconductor manufacturing plant.

J.T. Spierings

MN 420730

Master's thesis

Supervisor: prof. dr. ir. I.J.B.F. Adan
Advisors: dr. A.Y Pogromsky
dr. E.N. Ivanov (ASML)

EINDHOVEN UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF MECHANICAL ENGINEERING
MANUFACTURING NETWORKS GROUP

Eindhoven, June 17, 2013

FINAL PROJECT

EINDHOVEN UNIVERSITY OF TECHNOLOGY — April 2013
Department of Mechanical Engineering
Systems Engineering Group

Student	J.T. Spierings
Supervisor	Prof.dr.ir. I.J.B.F. Adan
Advisors	Dr. A. Pogromsky E.N. Ivanov (ASML)
Start	March 2012
Finish	March 2013
Title	Modeling the manufacturing performance of lithography systems in semiconductor fab environment.

Subject

In semiconductor manufacturing, model-based performance analysis is becoming increasingly important. Here, the performance is considered in a sense of throughput, cycle times and work-in-process. ASML produces lithography ('litho') systems for the semiconductor industry. Lithography is one of the most important processes in semiconductor manufacturing, and litho systems are the most expensive equipment in a fab. Litho systems are known to be the (designed) bottleneck, and they play a central role in wafer fabrication. Therefore, their performance has huge impact on the overall performance of the fab. Various parameters affect the performance: machine capacity, product mix, machine downs, repairs, dispatching policies, operator-induced delays, etc. Part of these parameters can be influenced by the design of ASML litho systems. Model-based analysis can be used to quantify the impact of these parameters on the overall performance of the litho area of the fab. The outcome of the analysis will be used to inform future design decisions and to help customers to improve the performance of their fabs.

Assignment

- Create discrete-event simulation model of networks of workstations in lithography area of semiconductor fabs, and to calibrate these models to data from the fabs of ASML customers.
- Review state-of-the-art literature on the topic and use ASML knowledge to identify the important input parameters for the model.
- Use the model and the knowledge to create a tooling that analyzes large amount of customer data and generates a performance overview with a breakdown of possible root causes of inefficiencies compared to theoretical achievable performance.
- Verify the model using actual data from a customer fab.
- Come up with improvement proposals for typical root causes of the cycle time inefficiencies in the litho area of the fab.

Prof.dr.ir. I.J.B.F. Adan — Dr. A. Pogromsky — E.N. Ivanov

Systems Engineering — Department of Mechanical Engineering

Preface

This is it, this chapter is the last chapter that I am writing for my master thesis. With this thesis I will finish my education and receive my Master of Science degree from Eindhoven University of Technology. This makes me both happy and sad. Happy because I will have finished my studies, which means I am ready to move on to my next challenge. But it also makes me feel a bit sad because I will leave a place that has given me so many great memories and taught me so many valuable lessons.

Most of these lessons were not part of the curriculum, but learned from the many activities that I shared with the friends and fellow students that I met during my studies. I would like to thank all of them for their love and support, and I am sure we will keep in touch.

During my master thesis I mostly worked at ASML, and I want to thank everybody in the Productivity group for making me feel at home right away and for providing me with your valuable advice. I also want to thank my fellow MN and SE students for being such great company during the time I spend in the lab.

I definitely want to thank my supervisors during this project: Evgeniy, for being my mentor on a daily basis, providing me with valuable input, arranging all the data that I needed for my research, and for keeping me motivated throughout the project. Sasha, for his valuable input, giving me the freedom to make this project feel like it was mine, and keeping my priorities straight at the same time. And, of course, Ivo for keeping his door open for me and taking the time to answer any questions I had.

Last but certainly not least I want to thank my family for their love and support. Without you this would not have been possible.

Enjoy reading my thesis!

Abstract

This thesis investigates how cycle time in the photolithography area of a semiconductor manufacturing plant can be improved. The research is focussed on two effects: The effect of unscheduled machine downs on the cycle times, and the effect of matching machines on cycle time. To conclude a tool set is developed to analyze any photolithography machine for inefficiencies and predict how much cycle time will be reduced if they are solved.

An aggregate simulation model is developed to investigate the effects of unscheduled machine downs and the effect of matching machines. The biggest improvement for unscheduled down is expected by focusing on reducing the longest repair times. Removing the top 10% repair times lowers the mean cycle time by 12%.

An even bigger improvement can be gained by matching two machines. If they are fully matched cycle times in the photolithography area can be reduced by up to 50%.

The aggregate simulation model that is created to find the effect of these improvements is extended to include more details from the photolithography area. Furthermore a GUI is added to automatically extract the inputs from log files. These can be used to analyse any photolithography machine and predict how much cycle time can be decreased if inefficiencies are removed.

Contents

Chapter 1

Introduction

The goal of this thesis is investigating how cycle time in the photolithography area of a semiconductor manufacturing plant can be improved. Cycle time is an important performance measure in semiconductor manufacturing [14]. It is the time it takes for a product to be processed, plus the time spent waiting to be processed. In a busy manufacturing plant the waiting time often greatly exceeds the processing time, products have to wait in a queue before the photolithography machine is available to process them.

These queues are there for a reason. There is always some variation in the time it takes for the photolithography machine to process a product. This means that the machine will sometimes take more time than the average to process a product, and sometimes less time. It is also possible that a number of short processing times follow each other. Having a queue of products in front of the photolithography machine means that there are always products available for the machine to process. Even if a sequence of short processing times occurs. This means that the machine will never have to idle, which is important because a photolithography machine is an extremely expensive piece of equipment. Prices for the newest generation of machines start at 30 million Euro. Having such an expensive machine sit idle would be a waste of capital.

The photolithography machine is often the most expensive piece of equipment in a semiconductor plant. This is why most semiconductor plants are designed to keep it processing as much of the time as possible: The photolithography area is designed to be the bottleneck. As such it always has a queue of products waiting to be processed. The bigger the variability in the processing time, the bigger the chance that a sequence of short processing times occurs, and the bigger the queue in front of the machine has to be to keep it from going idle.

Having queues also has a downside. The bigger the queues, the longer the waiting time for the products, and long waiting times mean long cycle times. Long cycle times, combined with variability, make it difficult to predict when products will be finished. Furthermore they lead to long times to market, long times before errors in the product are detected, and it causes a lot of money to be tied up in half finished products. Industrial studies [11] have shown that, on average, reducing the cycle time by one percent will decrease the cost per produced wafer by 0.7 percent. The photolithography cycle time is about 10% [1] of the total cycle time in a typical semiconductor manufacturing plant. This means reducing the cycle time in the photolithography cluster by one percent can reduce the overall cost per wafer by 0.07 percent.

The research presented in this thesis is done in corporation with ASML. ASML is a company that designs and builds photolithography machines and sells them to semiconductor manufacturing companies. It has been the leading supplier for photolithography equipment since 2002. Besides the current focus to design a new machine that can create smaller details, ASML is also interested in reducing cycle time for their customers. This thesis has three focus points to achieve this:

1. Investigate the effect of unscheduled machine downs on the cycle time for one photolithography machine.
2. Investigate how making machines work in parallel can improve cycle time in the photolithography area.
3. Develop tooling to analyze any photolithography machine for inefficiencies and to investigate how big their impact is on cycle time.

The first goal is to quantify the effect of unscheduled machine downs on cycle time. An unscheduled machine down occurs when a machine should be able to produce, but is not. This is typically caused by a part of the machine that breaks. The first part of the thesis investigates how changing the repair time for these unscheduled downs, as well as changing the frequency at which they occur, impacts cycle time.

The second part of the thesis focusses on a phenomena called machine matching. This is a complex procedure that makes it possible for products to be processed on two machines, instead of just one. It makes it possible for two machine to work in parallel, so if one machine can not process products for some time there is another machine that can take over part of the workload. Quantifying the impact of matching machines on photolithography cycle times is the second goal of this thesis.

The third and final goal of this thesis is to design and create a tool set that can automatically analyze a photolithography machine for inefficiencies and predict the impact of removing them, on cycle time.

1.1 Previous research

Previous research on similar topics can be found in a lot of sources. The most famous one is probably Factory physics [6], which gives a broad analysis of the behavior of manufacturing systems. Chapter eight [6] uses queueing theory to estimate the effect of different types of machine outages including unscheduled downs, however the analysis is limited to single machines subject to operation dependent failures. Adan and Resing [3] also use queueing theory to analyse manufacturing lines. Their work includes the analysis of a single machine subject to time dependent failure. The approaches in both of these works only apply to machines that process one product at a time. A photolithography machine consists of a series of processing steps, which means that multiple wafers can be processed at the same time. Because of this the results derived with queueing theory cannot be directly applied to a photolithography machine.

Creating an accurate simulation model that includes all the processing steps in a photolithography machine requires a lot of information and time. This can be overcome by using an aggregate simulation model. An aggregate simulation model combines everything that happens in a machine together into one effective process time (EPT). This approach uses the arrival and departure times for a machine to model it, instead of analyzing the internal workings like the queueing theory approach. Kock [9] and Veeger [21] used it to model photolithography machines. This resulted in very simple models that do not need a lot of data to generate accurate predictions for the mean cycle time. Including the overtaking behavior even makes it possible to predict the entire cycle time distribution [21].

At ASML, there have been previous studies by Van der Eerden [19], who used both a simulation model and queueing theory to investigate a theoretical photolithography machine and study the effect of rejects, rework, machine outages and tool dedication on both cycle time and throughput. During the same period Aarts [2] used a similar approach to study the effect of setup time, preventive maintenance and unscheduled downs on the cycle times for a theoretical photolithography machine. Both these studies used a detailed model that was not based on measured data, which makes it hard to translate their results to a real machine.

Babbs and Gaskins [4] analysed the effect of reduced equipment downtime variability on cycle time for semiconductor manufacturing by simulating an entire factory, based on universal SEMATECH data. They find a maximum reduction of 4.1% of cycle time if all variability is removed, which seems much lower than we would expect. They also investigate the effect of adding more machines to bottleneck stations, but in their case this was not the lithography area.

More general research on variability reduction was performed by Schoemig [15]. He investigated the effect on cycle time of reducing the variance of the time to repair. A theoretical production line, not specifically for semiconductor manufacturing, was simulated in order to do this. Unfortunately the results are not explicitly quantified as the focus is more on graphical inspection of the resulting cycle time - throughput curves. Taylor and Heragu [18] performed a similar study by simulating a number of different cases in a flow shop environment to find if reducing the mean or the variance of down time yields the biggest improvement on cycle time. They found that reducing the mean is usually more effective, but in specific cases reducing the variance can yield equal improvements.

Practical work was done by Van der Eerden et al [20], who performed a cycle time improvement study in an actual semiconductor manufacturing plant. They used a hybrid approach where EPTs are calculated from actual data from a factory operated by TI. Outliers in these EPT observation are analyzed and the effect on cycle times of removing them is calculated by using an extended Kingman equation. This leads to a list of possible improvements and their impact to the system. Their approach proved successful, after implementing the suggested improvements the photolithography cycle time was reduced by almost 50%. However, it is unclear how accurate their predictions were, but the results show that there can be a lot of potential for cycle time reduction in actual semiconductor factories.

In this thesis the EPT method will be used to create an aggregate model of a photolithography machine based on real data from a semiconductor manufacturing plant. The EPT method is enhanced to make it possible to explicitly model the effect of unscheduled downs on cycle times. This same aggregate model is extended and used to measure the effect of machine matching. For the final goal of this thesis a more detailed model is created that aggregates parts of the photolithography machine, to divide it into three parts.

1.2 Thesis outline

This thesis will start with a short introduction to semiconductor manufacturing. After this the approach to investigate the effect of unscheduled machine downs is explained, the data is analyzed and a sensitivity analysis is performed to find how unscheduled downs affect the cycle time for one photolithography machine. The same model is then extended to find the effect of matching on cycle time. Thirdly, a more detailed model for the lithography area is created. This model can be used to investigate the effect of changes to the inner workings of a photolithography machine. A graphical user interface is added to this model, together with scripts to automatically derive the required inputs to create a tool, to analyse the performance of any photolithography machine in a customer factory. The final chapter presents the conclusions from this thesis and recommendations for future work.

Chapter 2

Introduction to semiconductor manufacturing

Before the analysis of the photolithography area starts it is first important to know a little bit about the process of semiconductor manufacturing. This chapter will first give a short introduction of the entire process, followed by a more detailed explanation of the photolithography area in the second section.

2.1 Semiconductor manufacturing process

Semiconductor manufacturing is the processes of creating integrated circuits. These integrated circuits (IC) can have different functions, for instance a CPU or RAM to use in your mobile phone or PC. Every IC consists of a number of layers. The different processing steps required to build an IC are shown in Figure 2.1. Steps are repeated several times to form all the layers that will eventually form the integrated circuit.

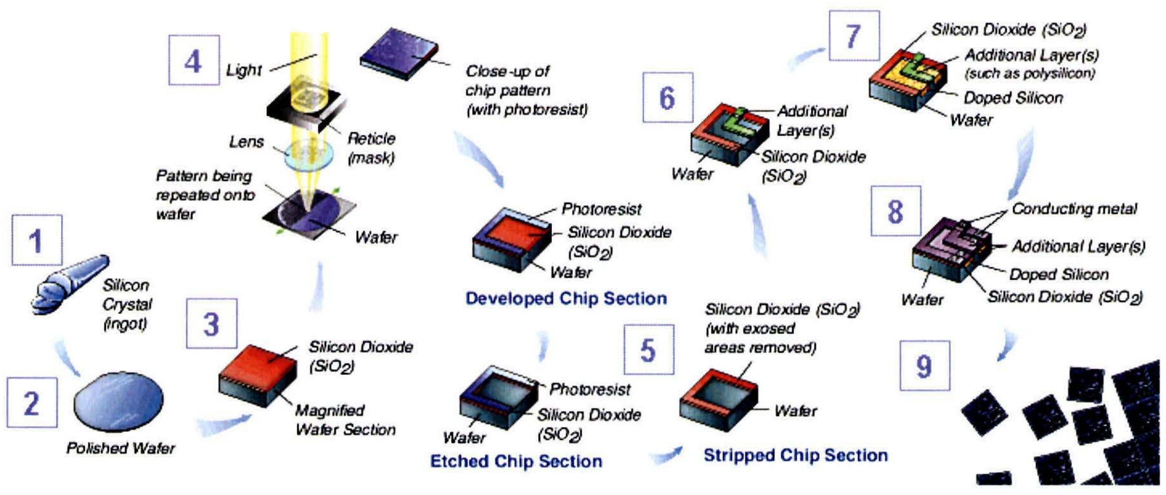


Figure 2.1: Processing steps in semiconductor manufacturing [16].

Each of the steps illustrate a specific step in the production process, and is explained below [16]:

1. Almost all of today's computer chips are built on silicon wafers. The first production step is to create these wafers by melting silicon. From this melted silicon single crystals are grown and sliced into flat cylinders, called wafers.

2. The second production step is polishing one side of each wafer to a mirror-like surface to remove all scratches and impurities. Chips are built on this surface.

3. In the third step a layer of silicon dioxide glass is deposited on the wafer. Because it will not conduct electricity this layer is called dielectric. A pattern to form the first layer is exposed on the wafer and etched to mask the silicon.

4. Step four is photolithography, or lithography for short. This is a process used to create patterns in the chip. First the wafer is coated with a light-sensitive chemical called photoresist. Then light is shone through a patterned plate called a mask or reticle to expose the pattern onto the resist, much in the same way film is exposed to light to form a photographic image.

5. Following the lithography process the wafer goes to step five, the etch area, where materials are removed using various manufacturing tools. Exposure to light in the lithography step caused the part of the resist that was exposed to "harden" (or become resistant to certain chemicals). The "non-hardened" resist is washed away in the development process. Then the material below it, for example SiO(2), is etched away by a biting fluid. The material that is covered by the resist is protected from this biting fluid. Finally the "hardened" resist is stripped off so that the remaining material underneath forms a three-dimensional pattern on the wafer.

6. The process of creating a three-dimensional surface on the wafer can be repeated to create complex structures.

7. The surface that was created in the previous steps is filled with chemicals that change its conductivity. Atoms from the chemicals called doping materials are diffused into the wafer through chemical exposure and heating. Dopant atoms displace some of the wafer's original silicon atoms to make the wafer either more or less conductive. This can also be done using ion implantation, which bombards sections of silicon with charged atoms called ions to displace silicon atoms.

8. Step eight shows the electro plating of the layer to form the chip's interconnections. A conducting metal (usually copper) is electro-plated on the entire wafer surface. Unwanted metal is then chemically and mechanically polished off to leave microscopically thin lines of metal interconnects in the three-dimensional structure. All the millions of individual conductive pathways in a chip must be connected using these metal layers, in order for the chip to function. This includes vertical interconnections between the layers as well as horizontal interconnections across each layer of the chip. Steps three to eight are repeated to form a number of layers that eventually for the chip.

9. Finally, after all the layers are finished each chip is tested for electrical performance. Any failing chips are marked so they can be discarded after all the chips have been sawn out of the wafer with special wafer saws. The chips are then put into individual

packages which will protect the chips and provide connections from the chips to the products for which they are designed. For example chips destined for computers are placed in packaging that can be plugged into computer circuit boards. The chips can be shipped to distributors. A finished wafer, before all the separate chips are sawn out of it, is shown in Figure 2.2.

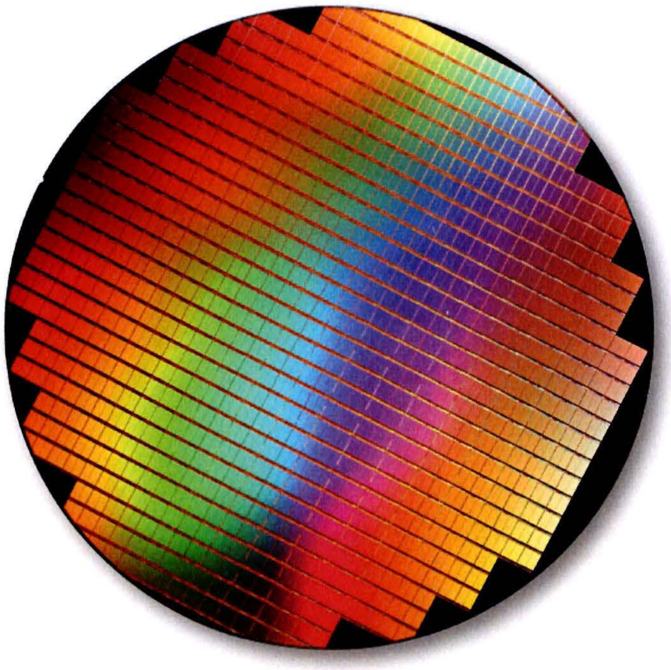


Figure 2.2: Finished wafer, ready for testing [12].

2.2 The lithography area

The lithography area, often further shortened to litho area, is the focus of this thesis. It consists of two machines: A resist track that applies the light sensitive layer to the wafers, and the scanner that exposes the pattern on the wafers. After the wafer has been exposed it will return to the track where it receives post exposure treatment. The track and scanner are physically linked together and effectively form one machine. This combination is sometimes called a litho cell. An actual litho area in a real factory is shown in Figure 2.3. The front of a number of litho cells can be seen on the left and right side.

The transportation of wafers to and from a typical litho cell is done in special containers, called Front Opening Unified Pods, FOUP for short. One FOUP can carry up to 25 wafers that are processed exactly the same. These FOUPs arrive to a buffer upstream of the litho cell. This buffer is called the stocker and it stores all the lots that are waiting to be processed, either on a litho cell or any other workstation. The transportation of FOUPs is automated in most fabs. A FOUP mounted on a litho cell is shown in Figure 2.4. In the rear a FOUP is being transported by the automatic transportation system.

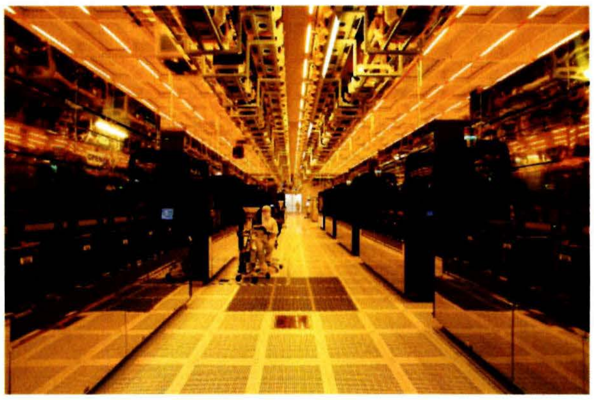


Figure 2.3: Litho area in an actual fab [12].

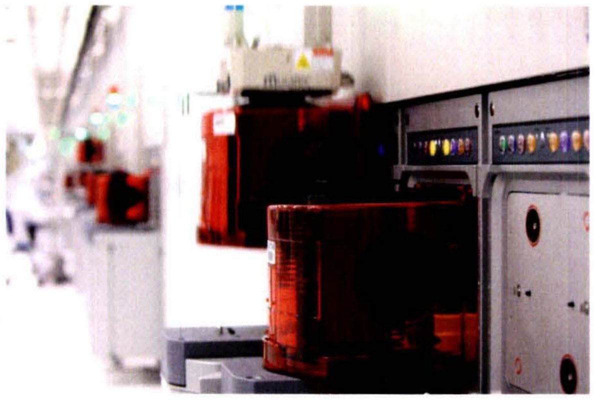


Figure 2.4: FOUP on machine [12].

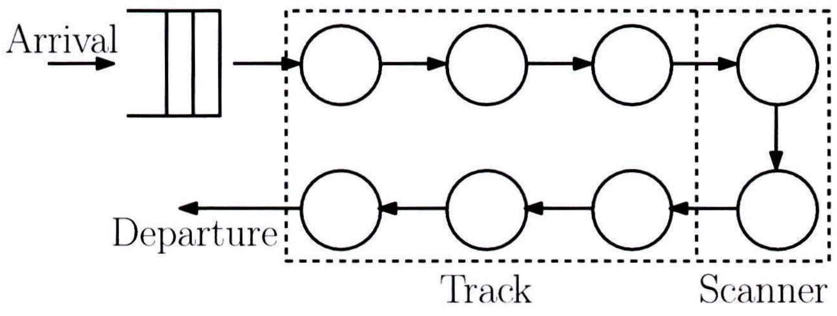


Figure 2.5: Schematic view of a litho cell.

A schematic view of one litho cell is shown in Figure 2.5. This shows a buffer and several processing steps in the track and scanner. FOUPs arrive to the buffer where they wait until a port on the litho cell is available. After this the wafers are loaded into the track to get the light sensitive coating, as explained in the previous section. The wafers then enter the scanner where they are measured and the image from the reticle is exposed. After exposure, the wafers receive a post exposure treatment in the track, are loaded into the FOUP again and leave the litho cell.

Explicitly modeling all the process steps in the track and scanner would require data for each of these steps. Most of this data is not available to us, making it impossible to create a model that describes all the processing steps in detail. Luckily for us, there are several methods to cope with this problem, and still create an accurate model. One of them, called the effective process time approach, is explained in the next chapter.

Chapter 3

Modeling approach

There are many ways to model systems like a litho cell. Data based models such as neural networks, analytical models based on queueing theory or discrete event simulation models are the most common approaches. The amount of data that is available for this thesis is not enough to build a data based model. We did not succeed in finding a queueing theory approximation in literature to describe a litho cell including machine downs, and deriving it ourself would result in an model that is probably to complex for ASML to maintain and use.

Discrete event simulation models, on the other hand, can be created even with very little data available and still produce accurate results. Especially aggregate models, that group a lot of events together to obtain effective process times of a litho cell, are a good example of this. Once a discrete event simulation model has been created, the process of extracting the inputs for this model from the data can be automated. This makes this approach well suited to reach the goals of this thesis.

In this chapter the modeling approach that is used to investigate the effect of unscheduled downs on on the litho cell cycle times is explained, and resulting model is presented.

3.1 Aggregate modeling using effective process times

Aggregate modeling using effective process times, EPTs in short, is a method to create aggregate models of a system. This method combines everything that happens during production into one process time. The result is that the actual production time is combined with all other effects that might occur while a product is being processed, like machine outages or an operator inspecting the product. It measures how long each products claims capacity from the machine. The advantage of this method is that a complex system like a litho cell can be modeled as a single machine, with the processing times equal to the effective process times [9, 21].

All that is needed to calculate the effective process times for a system are the arrival and departure times for products on the system. The algorithm to calculate EPTs is illustrated in Figure 3.1.

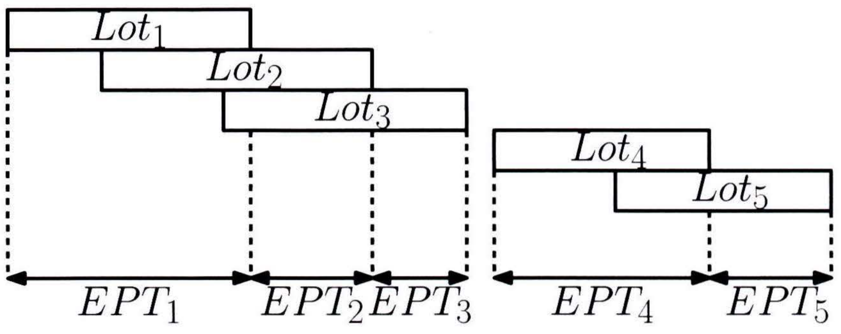


Figure 3.1: WIP dependency of EPTs.

A Gantt chart for five products is shown. The beginning of each bar indicates the arrival time, a bar ends as soon as the product leaves the system. The corresponding EPT observations are shown below the Gantt chart. An EPT observation is started if no other EPT observation is in progress and a product is present in the system. An EPT observation is ended as soon as a product leaves the system. The bars in Figure 3.1 can be divided into smaller bars to show all the processing steps in great details, but the EPT observations would be the same as everything is aggregated into one effective processing time. The algorithm to calculate EPT observations is summarized in (3.1).

$$EPT = d_i - max(a_i, d_{i-1}) \tag{3.1}$$

Here d_i is the i^{th} departure event and a_i is the i^{th} arrival event. Note that the i^{th} arrival and the i^{th} departure do not have to belong to the same product. In Figure 3.1 they do, but it is also possible that there are products that overtake one or more other products. If this is happens the i^{th} arrival and the i^{th} departure may belong to different products. If the machine is utilized 100%, (3.1) simplifies to the inter-departure times of the litho cell.

Overtaking is something that happens quite frequently in the lithography area. Modeling this behavior explicitly makes it possible to accurately simulate the cycle time distribution, instead of just the mean cycle time [21]. This behavior can also be calculated from the arrival and departure data. The order in which lots arrive can be compared to the order in which they leave, any overtaking can be seen if these two sequences are compared. The full algorithm to calculate the overtaking behavior can be found in Appendix D.

The EPT approach results in a model that works with a countdown timer and a queue. This countdown timer is used to simulate the effective process time. New products that arrive to the system are stored in the queue, and every time the countdown timer ends the first product in the queue leaves the system. The timer is started if a new product arrives and the queue is empty, or as soon as a product has left the system and there are still products in the queue. This is comparable to the way the EPT observations are calculated in (3.1). The EPT model and a schematic view of a litho cell are shown in Figure 3.2.

The EPT model also incorporates the overtaking behavior. Upon arrival a product is placed in the queue, but this does not necessarily have to be at the end of the queue. A product can overtake a number of products depending on the measured overtaking behavior in the actual system.

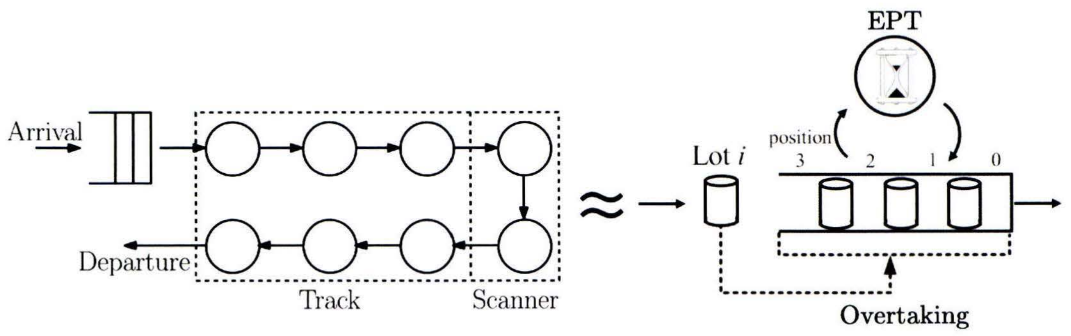


Figure 3.2: Litho cell and aggregate model.

One goal of this thesis is to quantify the effect of unscheduled downs on cycle times. The biggest advantage of using effective process times is that everything that happens between arrival and departure is combined into one processing time. Unfortunately this also means that unscheduled downs are combined with all other effects, which makes it impossible to analyze its effect on cycle time. To solve this problem a new method to filter unscheduled downs from the EPT observations is presented in the next section.

3.2 Modeling unscheduled downs

As mentioned before, the EPT approach aggregates everything that happens during processing into one effective process time. This is unfortunate, because we are interested in investigating the influence of unscheduled downs on cycle time. Before the EPT approach can be used for this, the effect of unscheduled downs has to be removed from the EPT distribution and modeled separately.

Two information sets are combined to do this: the EPT observations and the machine status. The EPT observations contain the beginning and ending time of each EPT observation. The machine status gives the state of the scanner for each moment in time. More information on these data can be found in Chapter 4. By combining these two data sets, it is possible to identify during which EPT observations an unscheduled machine down occurred. These EPT observations are then removed from the population.

After this, the "regular" behavior, without unscheduled downs, will remain. The effect of unscheduled machine downs is then reintroduced by modeling the failure behavior of the wafer scanner explicitly. This makes it possible to change the failure behavior and perform a sensitivity analysis. Unscheduled downs are modeled as preemptive resume interruptions, after the repair time has been simulated, the machine will continue production where it left of. This means that no products are scrapped. In reality there is some scrapping if there was an unscheduled down, but this is not always the case. Furthermore the effect of reworking the scrapped products on cycle times is assumed to be small, because failures do not occur very often and thus the number of extra products due to scrappage is small.

The approach of filtering EPTs based on the machine status is visualized in Figure 3.3. This figure shows Gantt charts of lots in a system, with the corresponding EPT observations below them. The machine status is shown above the Gantt chart as a colored bar. Green means the machine is up, red means the machine is down. It can be seen that the machine is down during the second EPT observation, which means that this observation will be removed from the population. The time it takes to repair the machine, and the time it takes for the machine to fail again after it has been repaired can be measured directly from the status data.

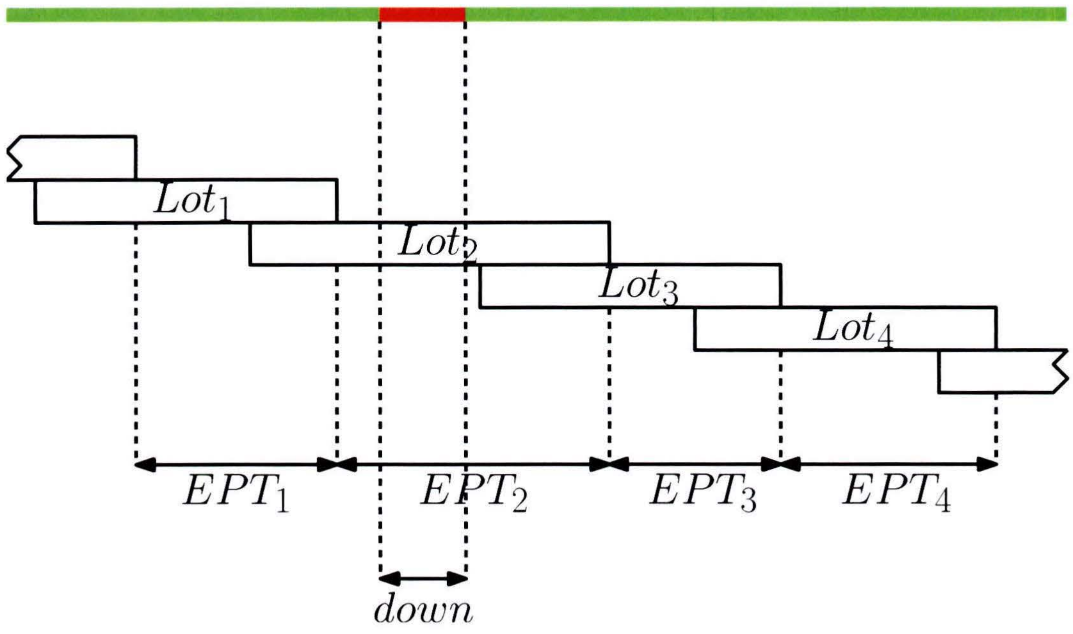


Figure 3.3: Combining EPT observations with status data.

The resulting model is shown in Figure 3.4. This model is similar to the model from Figure 3.2, except that there is an additional timer. This timer interrupts the EPT countdown timer to simulate the effect of unscheduled machine downs. This model is implemented in a discrete event simulation program, the details of this implementation are discussed in Appendix A. The inputs for this model are the EPT distribution, the arrival rate of new lots, the overtaking behavior, the times to repair and the times to failure. The output is a list of cycle times per lot.

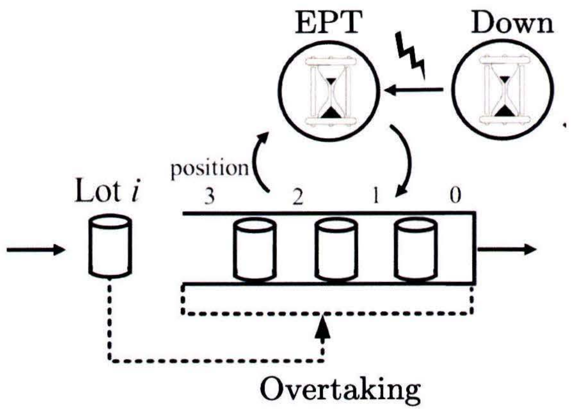


Figure 3.4: Aggregate model with explicitly modeled machine failures.

The EPT approach has two aspects that are important for this thesis, they are discussed in the following sections.

3.3 WIP Dependency

A litho cell is an integrated processing machine. There are several processing steps that happen inside a litho cell, which means that multiple products can be in process at the same time. For the system that is investigated in this thesis there can be up to five FOUPs in process simultaneously.

Previous studies [21, 9] have shown that this causes the EPT observations to be dependent on how heavily the litho cell is utilized. This effect is illustrated in Figure 3.5, which shows a Gantt chart for a fictional integrated processing machine that consists of two processing steps. Three scenarios are shown, one where the machine utilization is low, one where it is medium and one where it is high. Each scenario shows the corresponding EPT observations below the Gantt chart. It can be seen that as the machine utilization increases, more processing is in parallel. This causes in the EPT observations to be shorter as the utilization increases.

A common solution for modeling this utilization dependency is to use the number of lots that are present at the start of an EPT observation to divide the observations into buckets. This approach leads to a work in process (WIP) dependent EPT distribution. Every time a new EPT is started in the model, the WIP level is checked and a new EPT is sampled from the appropriate WIP bucket. Others have done research on different methods to model this utilization dependency [7], but this thesis will use the conventional WIP buckets as it has been proven to work sufficiently accurate for semiconductor manufacturing environments.

3.4 Traceability

Outliers in the EPT distribution tend to be caused by inefficiencies or other hiccups in the litho cell. This means that finding the root causes for these outliers and solving them can improve the system. The identification of these root causes is not as straightforward as one might think. EPT observations can contain processing time for more than one lot if there is overtaking in the system. This makes it difficult to match specific lot properties, like the number of wafers or the recipe, to an EPT observation. This effect is illustrated in Figure 3.6, which shows the Gantt charts of four lots that are processed according to a preemptive last in, first out policy. The corresponding EPT observations are shown under the Gantt chart.

It can be seen that the first EPT observation includes processing time for the first, second, third and fourth lot. The second EPT observation contains only process time for the third lot, but not all of it since part of the processing already happened during the first EPT observation. The same holds for the third and fourth EPT observations. In this case it is difficult to match the EPT observations to specific lots, which can make it a bit difficult to find root causes for outliers.

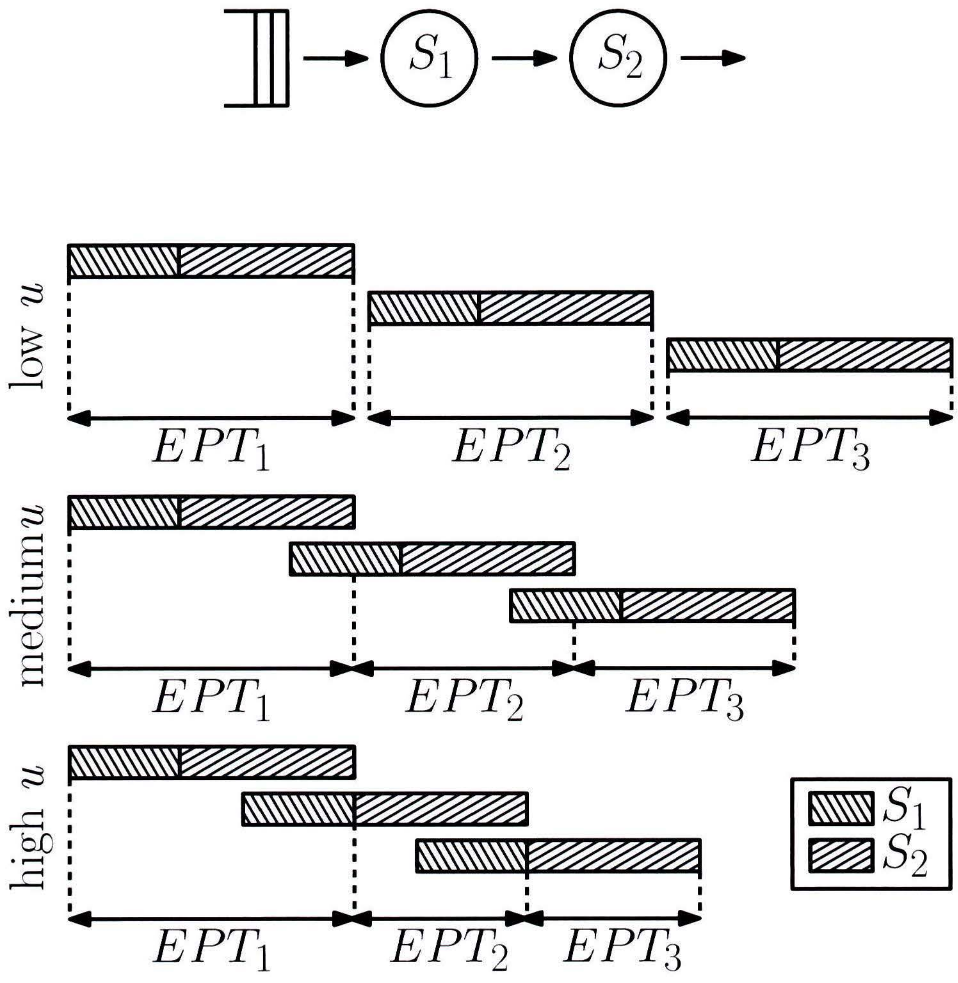


Figure 3.5: Utilization dependency of EPTs.

A possible solution could be to match the attributes of all lots that are in the system during an EPT observation, to this observation. However, this might result in an overload of attributes for EPT observations during high utilization. In this thesis we combine the machine state to identify outliers that occur during unscheduled downs, and we investigate the Gantt chart of the actual lots to investigate the root causes for other outliers.

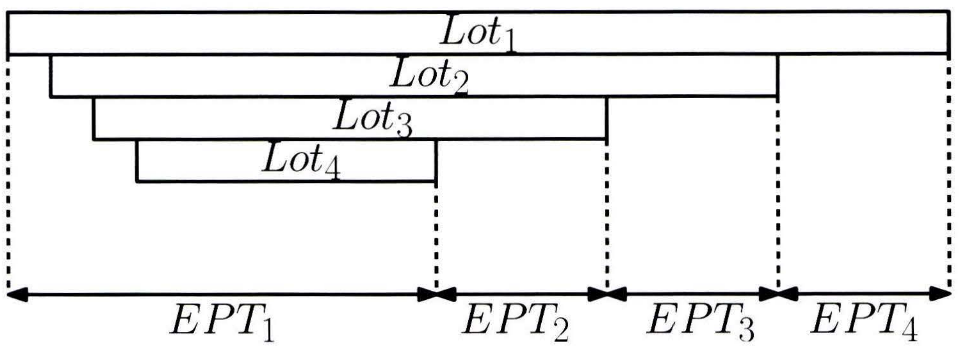


Figure 3.6: Overtaking makes it difficult to match EPT observations to specific lots.

Chapter 4

The data

The EPT model that is used in this thesis requires a number of inputs to generate an accurate cycle time for each lot. The required inputs are the inter-arrival time distribution, the EPT distributions for each WIP level, the overtaking distributions for each WIP level, the time to repair distribution and the time to failure distribution. These inputs are calculated from data of an actual semiconductor manufacturing plant. The following three data sources are used:

- Manufacturing Execution System data for one litho cell.
- Scanner status data for the same litho cell.
- Unscheduled down data for all scanners of the same type worldwide.

The Manufacturing Execution System (MES) data contains information about each lot that was processed on the litho cell that is studied in this thesis. This information includes general information about the lot, like the number of wafers or the recipe that was used, as well as the times the lot entered and left the different processing steps in the litho cell. The arrival and departure times to and from the litho cell are used to calculate the EPT observations and overtaking behavior, as well as the inter-arrival times.

The machine status data stores the state that the scanner was in. This file is combined with the EPT observations to identify and remove the EPT observations that contain an unscheduled down (USD) of the scanner, as shown in Figure 3.3. The MES file and the status file both span the same time period of approximately six weeks.

The status data can also be used to calculate the time to repair (TTR) and the time to failure (TTF) for our scanner. Unfortunately the six weeks of data does not contain enough machine failures to properly model the time to repair and time to failure. This is where the third data source comes in. This file contains information about the unscheduled down (USD) behavior of every scanner of the same type that our data is based on, that was in use during the last year. This data does provides us with enough data to properly model the time to failure and time to repair.

The data and how it is used is shown schematically in Figure 4.1. The following sections explain the three data sources in more details.

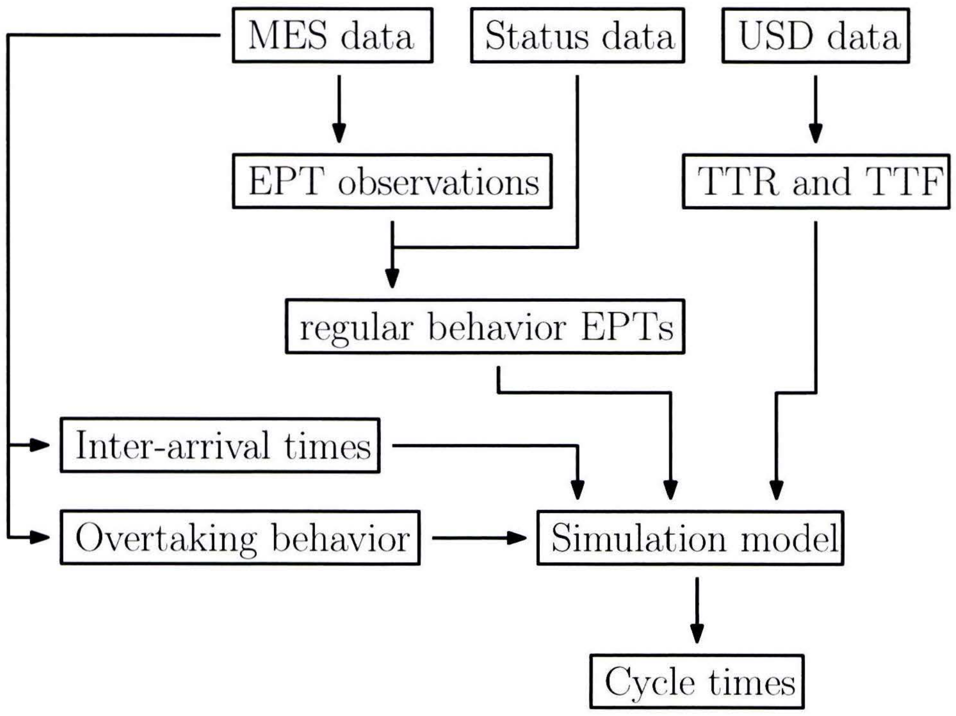


Figure 4.1: Flow of data from input files to simulation model.

4.1 MES file

The MES file contains information about approximately 3000 lots that have all been processed on the same track and scanner combination, and spans a time of approximately six weeks. All available information for each lot is stored in one row, and every column contains a different piece of information. The most useful information is shown in Table 4.1. For now, only the time stamps for *Left previous processing step* and *Depart from port* are used as the arrival and departure times to calculate the EPT observations. A more detailed model is introduced in Chapter 10, which needs some additional attributes to calculate the required EPTs. This section continues to explain the attributes that are mentioned in Table 4.1.

Attribute	Data type
Lot ID + counter	Number
Port	Number
Number of wafers	Number
Recipe	String
Left previous processing step	Time stamp
Arrived on port	Time stamp
Start loading	Time stamp
Finished loading	Time stamp
Depart from port	Time stamp

Table 4.1: Most important content in MES file

The *Lot ID* is the name of the FOUP with its wafers. It is used to distinguish between different FOUPs. A FOUP can visit the machine multiple times for different layers. The counter shows how many times the lot has visited the litho cell within the time span of the data. By adding this to the lot ID a unique entry is guaranteed.

The *port* shows which of the five ports was occupied by the FOUP.

The *number of wafers* shows how many wafers were in the FOUP. This number can range from 1 to 25. Production FOUPs usually carry between 20 and 25 wafers. Besides production FOUPs there are also monitor FOUPs, these are used to monitor the machine and check if it is still producing according to specification. Monitor FOUPs usually contain a smaller number of wafers.

The *recipe* shows which recipe was used to process the wafers. Each recipe has specific settings for each of the steps in the track, like pre-bake time and temperature, and specific exposure specifications like exposure time and which image has to be exposed.

The *left previous processing step* time stamp is the time the FOUP departed from the upstream machine. This is the best available estimate for the time the lot arrived in the stocker. It includes a small amount of extra time to be transported from the previous machine to the central stocker.

The *arrived on port* time stamp is the time the FOUP was physically mounted on the machine and is ready to start being loaded into the track.

The *start loading* time stamp is the time the wafers in the FOUP started to be loaded into the track. This time is equal to the time at which the wafer handler system begins to transport the first wafer from the FOUP.

The *finished loading* time stamp is the time the last wafer was loaded into the track.

The *depart from port* time stamp is the time the FOUP was transported from the litho cell to be processed on the next machine. This time is used as the time the lot finished processing on the track-scanner combination. This also includes some additional waiting time for the transportation system.

It is assumed that all the time stamps in the MES file are logged using the same clock, so there are no synchronization errors in the data. It is unclear if this is actually the case in reality. If there are any deviations however, their effects are likely to be small on the scale of cycle times.

4.2 Status file

The status file contains the status of the scanner during the same six weeks as the MES file. Note that this file does not include the status of the track that is attached to the scanner. Any errors in the track can therefore not be identified with this data. The status file gives the state of the scanner at any time, this can be any of the following four states:

- Productive
- Idle

- Scheduled down
- Unscheduled down

The *Productive* state means that the scanner is processing a lot. *Idle* means the scanner is idle but ready to start processing a new lot. *Scheduled down* means the scanner is undergoing planned maintenance, and can not process a new lot until this is finished. The fourth state, *Unscheduled down*, means that the scanner is not processing any new lots because it is down due to an unexpected error. It can not process any new lots until this error is resolved. Any EPT observation that contains an *Unscheduled down* is removed from the population, the other three states are left in.

The status file is assumed to be synchronised to the MES file. Any deviation here could cause EPT observations to be labeled incorrectly which might lead to the incorrect removal of them.

4.3 USD file

The third data source is a collection of unscheduled down starting times and durations, during one entire year, for all the machines of the same type as the machine that was used in the previous two data sources. This file only includes information about unscheduled downs. Machine failures happen at a relatively low frequency. This means that the six weeks of data that is available in the status file is not enough to correctly model the failure behavior. Using an entire year for multiple machines does make this possible.

Unfortunately there is a problem with the method that unscheduled downs are entered into this file. This problem is presented and solved in the following section. Besides this, it is important to know if the failure behavior is time dependent, or operation dependent. This is investigated in Section 4.3.2.

4.3.1 Time to failure reconstruction

The unscheduled down logs that comprise the USD file are created as the local ASML team repairs the scanner in a customer factory. Sometimes a machine is assumed to be repaired, but after testing it turns out there is still an error. In such a case multiple attempts are needed to fix a machine, which means that one unscheduled down is logged as multiple smaller ones. This is seen in the data as multiple unscheduled downs that occur very shortly after each other.

This effect is corrected by comparing the time to failure to the average processing time that was found in the MES data, and combining machine downs if the time to failure is smaller than the average production time. This method is based on the fact that the local ASML team will stay with the machine until it is successfully producing again. If the time to failure is smaller than the average production time for one FOUP, it is very likely to be caused by the same error that was not resolved in the previous attempt.

This method is illustrated in Figure 4.2. The top of this figure shows a situation with three unscheduled downs. The first two are separated by a time that is smaller than the average

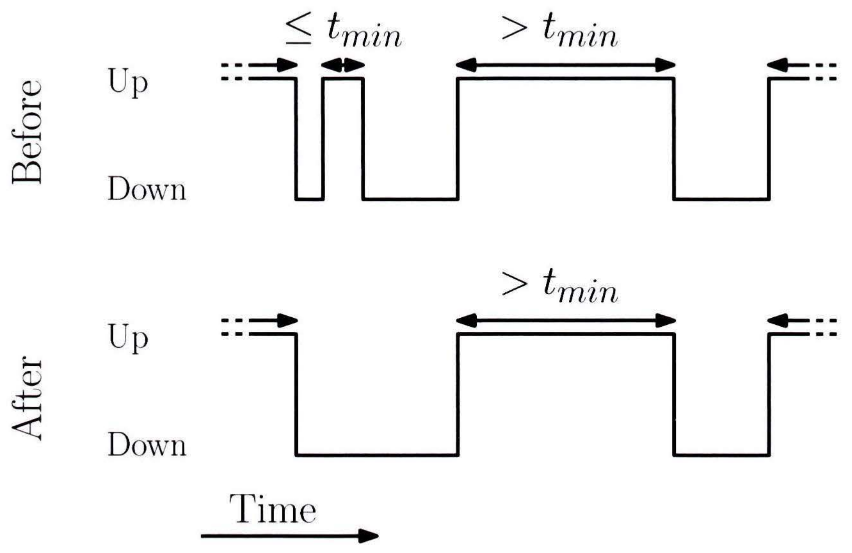


Figure 4.2: Flow of data from input files to simulation model.

production time, t_{min}. After the proposed correction these two unscheduled downs are concatenated into one down, including the uptime in between. The algorithm to perform this correction is shown in Appendix F.

The original time to failure and time to repair distributions are shown in Figure 4.3. A large number of small times to failure and small times to repair can be seen here. The time to failure and time to repair after the correction are shown in Figure 4.4. It can be seen that the number of short time to failure observations is decreased significantly, as expected. The number of short time to repair observations is also decreased significantly which can also be expected as multiple short times to repair and times to failure are concatenated into one big time to repair. The resulting distribution shapes look more like exponential distributions after the correction. This seems like a nice result, but note that this was not the goal the corrections.

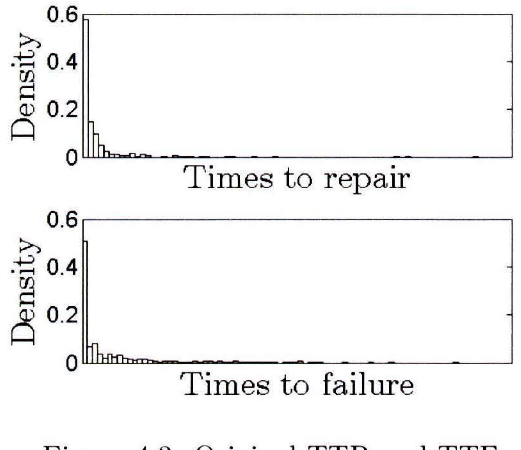


Figure 4.3: Original TTR and TTF.

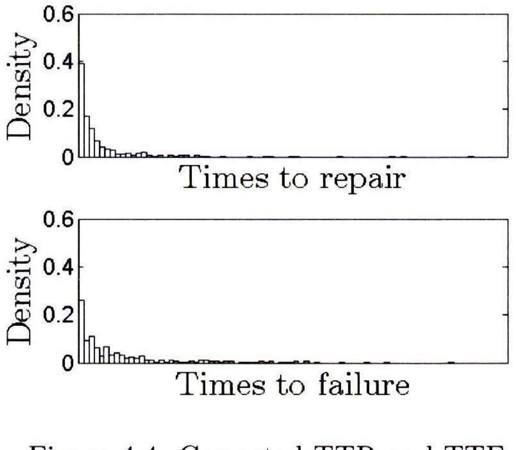


Figure 4.4: Corrected TTR and TTF.

4.3.2 Time dependent failure or Operation dependent failure

In the literature there are mainly two ways to model machine outages: operation dependent or time dependent. Before a simulation model can be created, it is important to know which of these approaches best describes the actual situation.

Operation dependent failures assume that a machine can only break if it is processing a product. The failures are then typically caused by wear in the machine.

Time dependent failure, on the other hand, assumes that a machine can break down at any time, even if it is idle. A machine that fails while it is not processing might seem like a strange phenomena but in reality such a situation can be caused by a number of sub systems that are always on, like the electronics.

ASML has information about average utilization and the mean time to repair (MTTR) for a number of machine, similar to the one that is investigated in this thesis. Figure 4.5 shows a plot of this mean time to failure against the utilization for each of these machines.

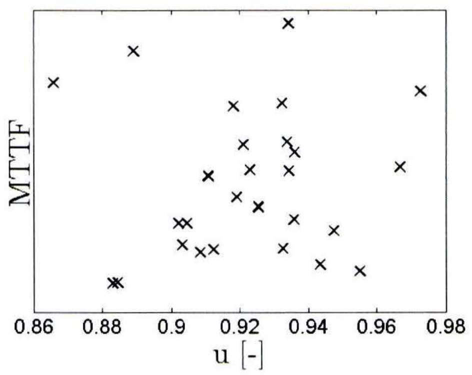


Figure 4.5: Mean time to failure as a function of machine utilization.

If the failure behavior was operation dependent this would mean that as the machine is utilized more, the time difference between failures gets smaller. In other words, if the machine is used heavily this will lead to more failures in the same time period than if the machine was only used lightly. No such trend is visible in Figure 4.5, which suggests that the failure behavior is time dependent.

4.4 Missing entries

The MES file that is used to calculate the inter-arrival times, EPT observations and overtaking behavior is not flawless, some entries are missing. The effect of these missing entries is explained in this section.

Most of the missing entries are near the beginning and ending of the MES file. Lots at the beginning of the file are missing their arrival times, because they arrived before the starting time of the file. The same holds for the end of the file, where lots are missing departure times because they leave the system after the ending time of the file. The calculations of the inter-arrival times, EPTs observations and overtaking behavior are only based on lots that have both an arrival and a departure time stamp.

A benefit of this choice is that the number of inter-arrival times, EPT observations and overtaking observations is the same as the number of measured cycle times from the data. This makes it easier to check our input parameters for correlations.

The lots that only have a departure time at the beginning of the file could still be used to calculate EPT observations, although it would not be possible to determine the corresponding WIP levels because the arrival times are unknown. The lots near the end of the file that only have an arrival time could be used to calculate additional inter-arrival times. These extra inter-arrival times are not needed in our case, because the number of inter-arrival times from complete lots is already large enough to properly identify the arrival behavior.

Not taking the incomplete entries into account means that the WIP level is not always estimated correctly, especially for the first couple of EPT and overtaking observations. The lots near the beginning of the file that only have a departure time stamp can be used to estimate the WIP level for this first part. Their arrival time could be manually set to earliest time that is present in the file. However, this would result in a number of incorrect EPT observations that would have to be identified and filtered out again. The effect of underestimating the WIP level is assumed to be small because the total number of EPT observations is large.

Only using complete entries reduces the time frame of the MES file from six to approximately four weeks, which is assumed to be long enough to properly model the inputs.

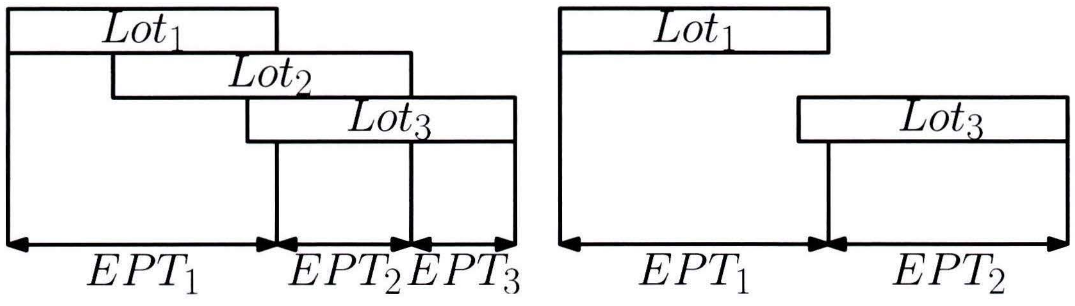


Figure 4.6: Calculating EPTs with missing data.

Missing entries at the beginning and end of the MES file is something that can be expected. However, some entries in the middle of the file are also missing time stamps. These time stamps are replaced by a question mark and usually occur when there is a failure in either the scanner or the track.

Missing entries in the middle of the data leads to an overestimation of the EPT observations. This effect is illustrated in Figure 4.6.

The left Gantt chart shows three lots that arrive to a system and depart again some time later. The right Gantt chart is the same as the left one, except that the second lot is missing. The corresponding EPT observations are shown below the Gantt charts and it can be clearly seen that they are different because the second and third EPT are now combined. This effect is assumed to be small as the number of missing entries in the middle of the file is small. Furthermore, part of these overestimated EPT observations is already removed because they contain unscheduled machine downtime.

Chapter 5

Input analysis

Analysis of the inputs is an important step to create an accurate simulation model. If there are any trends or dependencies in the input data, it is important to include these in the simulation model. In this chapter the inter-arrival times, EPT observations and overtaking behavior is derived from the data sources that were explained in Chapter 4. Each of them is then analyzed as follows:

1. Exploration: The data is visually inspected for any outliers or other deviations.
2. Trends: The data is checked for any first order trends to see if it is increasing or decreasing with time.
3. Autocorrelation: The data is checked for any other patterns or dependencies with itself.

Based on these analyses a discrete event simulation model has been created. This simulation model is validated in the last section of this chapter. Some of the figure axis in this chapter are left blank due to confidentiality.

5.1 Inter-arrival times

The inter-arrival times are the first input parameter that is calculated. They are calculated as the difference between subsequent arrivals to the system. The formal definition is shown in (5.1).

$$ta_i = \tau_{i+1} - \tau_i, \tag{5.1}$$

where ta_i is the i^{th} inter-arrival time, τ_i is the i^{th} time a lot arrives to the system. The resulting inter-arrival times are analysed in the following subsections.

5.1.1 Exploration

The first step to analyse the inter-arrival times is exploring the data. A scatter plot of all the inter-arrival times is shown in Figure 5.1. There are a four values that are significantly bigger than the rest around days 18 and 25. Around these two days two long unscheduled downs occurred, and the arrivals to our machine were halted. This also causes the gap around day 18. More details on this are explained in Section 5.4. The resulting scatter plot is shown in Figure 5.2. There are no further deviations in this scatter plot.

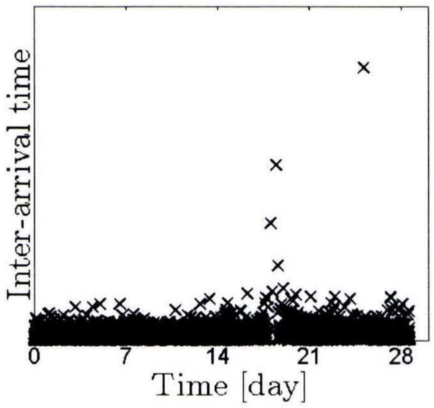


Figure 5.1: Inter-arrival times.

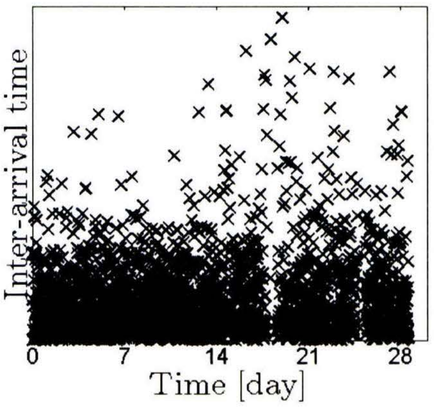


Figure 5.2: Inter-arrival times, no outliers.

5.1.2 Trends

After removing the outliers from the data it is important to check if there is any trend in the inter-arrival times. A least squares line is fitted on the data without the outliers, to check if the data might be increasing or decreasing over time. The result is shown in Figure 5.3.

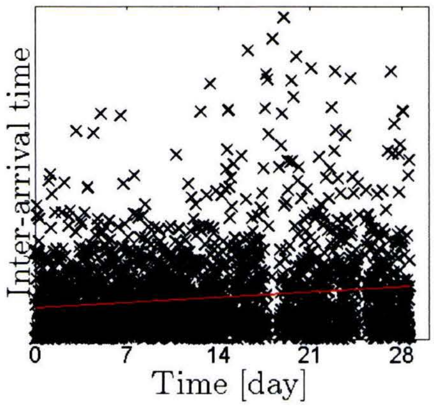


Figure 5.3: Inter-arrival times with trend line.

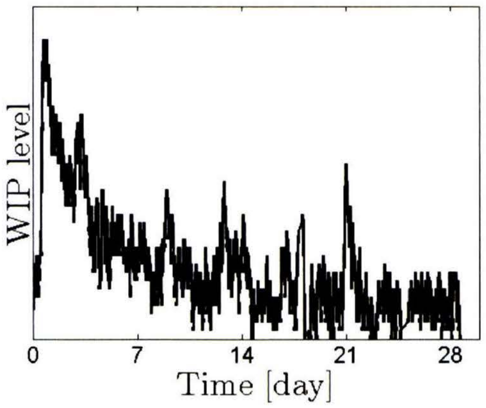


Figure 5.4: WIP level over time.

A significant upwards trend can be seen in the least squares fit. This indicates that the inter-arrival times are increasing over time. This same trend can also be seen in the WIP level, which decreases over time as shown in Figure 5.4. It is possible to include this trend in the simulation model, for instance by making the arrival process time dependent.

However, we plan to predict the effect of machine outages. This effect will be bigger if the utilization is high, and smaller if the utilization is lower. For the analysis a fixed utilization level is used, which means that incorporating the trend in the inter-arrival times in the simulation model, is not needed. The only reason to need this trend in the simulation model is to validate if it can predict cycle times accurately at the training point. This is done by using the actual stream of inter-arrival times, as measured in the data.

5.1.3 Autocorrelation

The third thing to check is whether the inter-arrival observations are independent of each other. There might be other dependencies in the inter-arrival times besides the trend that was found in the previous section. The sample autocorrelation factor is used to check this. This factor is defined as $r_k = \frac{c_h}{s^2}$, where s^2 is the sample variance and c_h is the covariance between inter-arrival observation at different times. This time difference is called the lag. The sample covariance function for a discrete series is shown in (5.2).

$$c_h = \frac{1}{N-h} \sum_{i=1}^{N-h} (X_i - \bar{X})(X_{i+h} - \bar{X}), \tag{5.2}$$

where h is the lag, N is the total amount of observations and X is the set of all observations. Dividing the covariance by the sample variance results in a value between -1 and 1, where a value of 1 means high positive correlation, -1 means high negative correlation and a value of 0 can be an indication that there is no correlation. The sample autocorrelation factor is plotted for up to 40 lags. The results are shown in Figure 5.5.

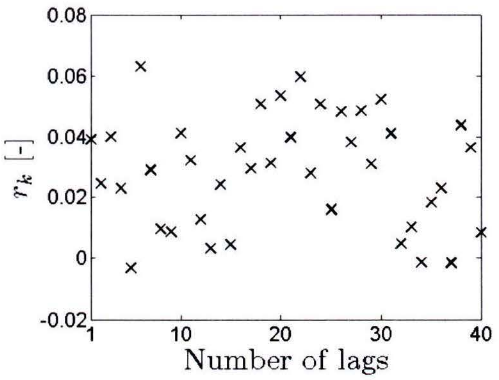


Figure 5.5: Autocorrelation factor for inter-arrival times.

It can be seen that the autocorrelation is slightly positive for all amounts of lag shown in the plot. This is expected because of the positive trend that was found in the inter-arrival data in the previous section. There do not seem to be any other dependencies or patterns in the data. Experiments were performed with more lags and the results were the same.

Based on these analyses the inter-arrival times can be modeled by a stationary process with independent sampling from the inter-arrival observations, to measure the effect of unscheduled downs. To validate the simulation model the original stream of inter-arrival times is used.

5.2 EPT observations

The EPT observations are calculated using the algorithm in Appendix D. In the following subsection these observations are analyzed to find any outliers and other irregularities, linear trends and other dependencies or patterns. Based on these analyses the proper way to model the processing times in the simulation model is determined.

5.2.1 Exploration

The EPT observations and their ending times are shown in Figure 5.6. Any red EPTs in this figure have overlap with an unscheduled down. The rest is plotted in black.

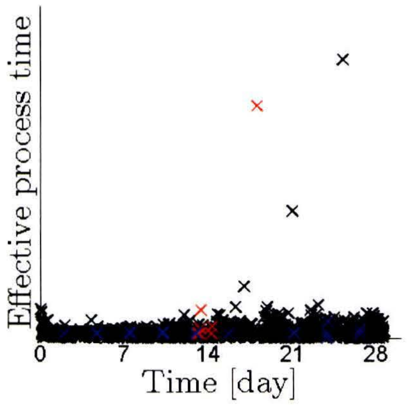


Figure 5.6: EPTs vs. time.

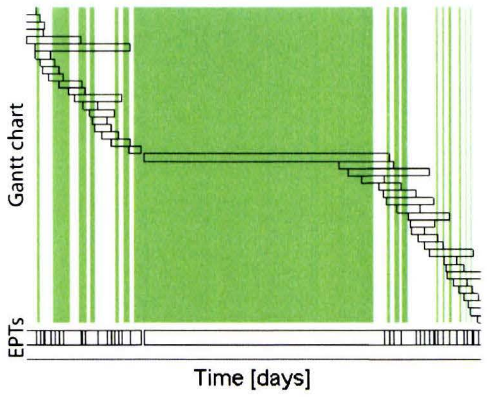


Figure 5.7: Third unidentified outlier.

There are four values that are far bigger than the rest of the population, and only one of them is identified by combining the machine state and EPT observations. These EPT observations most likely do not belong to one specific lot, as explained in Section 3.4. To find out what happened with the other three outliers the actual arrival and departure times for the lots, the machine state and the EPT observations are inspected. They are all plotted together for each of the unidentified outliers. Figures 5.7, 5.8 and 5.9 show the largest, the second largest and the smallest unidentified outliers respectively. The status is shown as the background color, where green means that the scanner is idle, white means the scanner is processing and blue means the scanner is scheduled down. The EPT observations are shown below the Gantt charts.

The figures show that all unidentified outlier are observed during a long scanner idle. After inspecting the MES file for the lots that are in the system around the time of the outliers we find that there is a big time difference between the *Left previous processing step* time stamp and the *Arrived on port* time stamp. This indicates that there are long periods of time during which no lots get dispatched around the time that the outliers occur.

This dispatching delay lasted 11 hours for the biggest outlier. Looking at the Gantt chart in Figure 5.7 shows that there are no lots arriving to the system anymore, and there is just one lot in the system. The fact that the scanner is idle during this time can indicate a number of

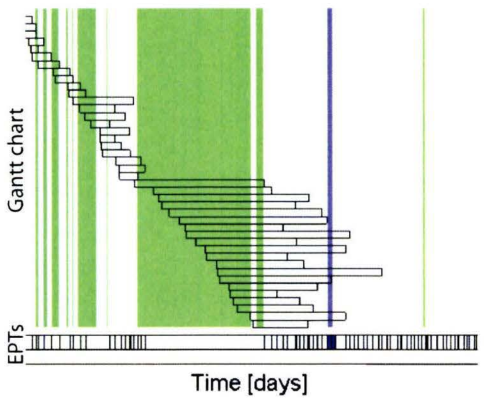


Figure 5.8: Second unidentified outlier.

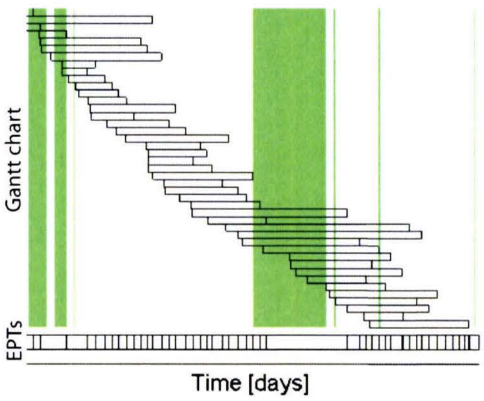


Figure 5.9: First unidentified outlier.

causes: The track or transportation system can be down (remember that we only have status data for the scanner) so arrivals to the system are halted and the lot that is on the litho cell cannot leave. Another cause can be that WIP in the factory was low and this last lot was on hold, waiting for an operator to inspect it.

The dispatching delay for the second biggest unidentified outlier was six hours. The Gantt chart in Figure 5.8 shows that lots keep arriving to the system this time, but the scanner status is still idle. This suggests that the production might have been put on hold temporarily. This is sometimes done to prevent WIP from piling up on the downstream machine [5, 10]. Another reason can be that there was a failure in the track which causes the scanner to become idle.

The smallest of the three unidentified outliers is shown in Figure 5.9. Again, a big delay is found in the MES file before lots are being dispatched. This delay lasts five hours for this outlier. The corresponding Gantt chart looks similar to the second biggest outlier: Lots keep arriving but no lots are leaving the system. The scanner is idle again for quite a long time.

Based on this information we decide that all three of these unidentified outliers are not part of the "regular" processing behavior and they are removed EPT population. The EPT observations during an unscheduled machine down are also removed from the population, because they will be modeled separately. The remaining EPT population is called the "regular effective process times". This population is shown in Figure 5.10 against the observation time. The EPT observations for integrated processing machines tend to be dependent on the machine utilization level, as explained in Section 3.3. Figure 5.11 shows the EPT observations plotted against the WIP level when the observations started. A clear dependency on the WIP level can already be seen here.

The resulting observations are still quite noisy. To make the WIP dependency more clear the mean, coefficient of variation and the number of EPT observations per WIP level are shown in Figure 5.12. This figure shows that the mean and squared coefficient of variation do not change much for WIP levels eight or higher. A saturation level is expected because at some point there are no more processing steps that can be performed in parallel. However,

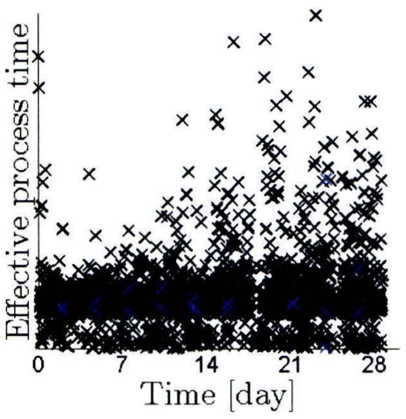


Figure 5.10: EPTs vs. time.

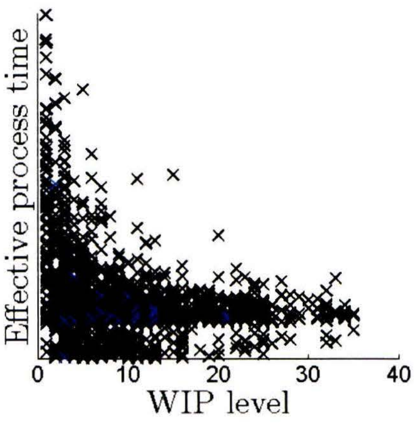


Figure 5.11: EPTs vs. WIP level.

we would expect this to happen at a WIP level of five, because the physical system can only process a maximum of five FOUPs simultaneously. These extra three FOUPs might give more options to the computer that selects which FOUPs to dispatch to the machine, so it might be able to select more similar lots, which can lead to more efficient cascading. This will result in smaller setup times, and thus smaller processing times. Based on Figure 5.12 the EPT observations are divided into eight WIP buckets: levels one to seven, and eight or more. These eight WIP buckets are explicitly coded in the simulation model to correctly determine the processing times.

5.2.2 Trends

The fact that the EPT observations are negatively correlated to the WIP level in the system means that they will increase over time in our data. This is because the WIP level in the system decreases over time, as shown in Figure 5.4. This trend is already taken into account in the simulation model. However, we are still interested in analyzing the EPTs in each of the eight WIP buckets for trend separately.

Figure 5.13 shows the scatter plot for the EPTs in each of the WIP buckets. Similar to the inter-arrival times, a linear least squares line is fitted in each plot.

It can be seen that the trend line for the first WIP bucket decreases over time. This trend is caused by a single observation at day zero, after which there are no observations for quite some time. The WIP level for observations around day zero is possibly inaccurate because any rows with missing entries are discarded, as explained in Section 4.4. The EPTs in the other WIP buckets do no show any clear trend, as their trend lines are nearly horizontal. This suggests that the EPT observations for each of the WIP buckets can be modeled as a stationary process.

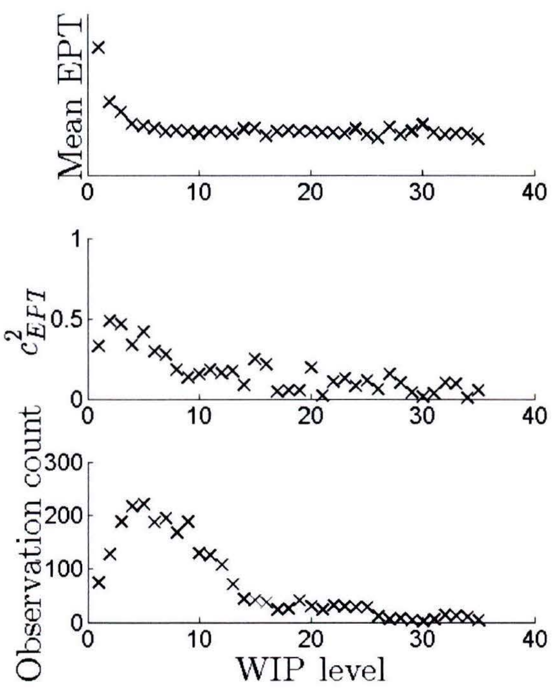


Figure 5.12: Mean, coefficient of variation and count of EPT observation per WIP level.

5.2.3 Autocorrelation

The final check for the EPT observations is to see if there are any other patterns in the regular EPT observations. The autocorrelation coefficient is used again to investigate this. This coefficient is calculated for different lags, for each of the WIP buckets separately.

The results for up to 40 lags are shown in Figure 5.14. It can be seen that for the first seven WIP buckets there is a large amount of spread in the autocorrelation factor. However, the values are all spread around zero which leads us to believe that there is no autocorrelation in the data. The spread is most likely due to the relatively small sample sizes for the first seven WIP buckets.

The highest WIP bucket of eight or more contains significantly more observations and it can be seen that there is less spread, and also that the overall values are very small for up to 40 lags. Tests were done with more than 40 lags and the results were the same. These results suggest that the EPT observations are independent of each other, which means that the processing times in the simulation model can be described by identically independent sampling from the appropriate WIP bucket.

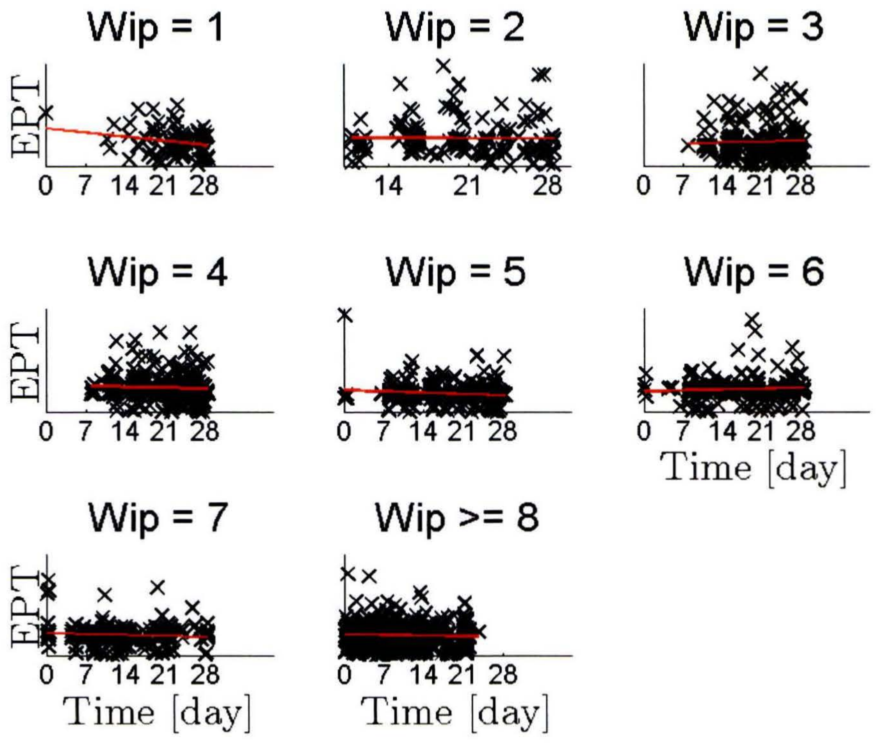


Figure 5.13: Regular EPTs divided into WIP buckets, with trend lines.

5.3 Overtaking behavior

The overtaking behavior is calculated together with the EPT observations, using the algorithm shown in Appendix D. The overtaking behavior will be inspected in a similar fashion as the inter-arrival times and the EPT observations.

5.3.1 Exploration

A scatter plot of the number of overtaken lots is shown in Figure 5.15. The number of overtaken lots is also plotted against the WIP level, as shown in Figure 5.16. The number of observations per WIP level are shown in Figure 5.17. There do not seem to be any outliers or other irregularities in the data besides the WIP dependency. This WIP dependency can be explained by the fact that as there are more lots in the system, more lots can be overtaken. The decrease in WIP level over time can also be found in the overtaking behavior, due to the WIP dependency and because of this further analysis of the overtaking behavior is done for the separate WIP levels. The number of observations is low for most WIP levels, except those between two and ten. Because of this, further analysis of will only be done using the observations for these WIP levels, where there are enough observations.

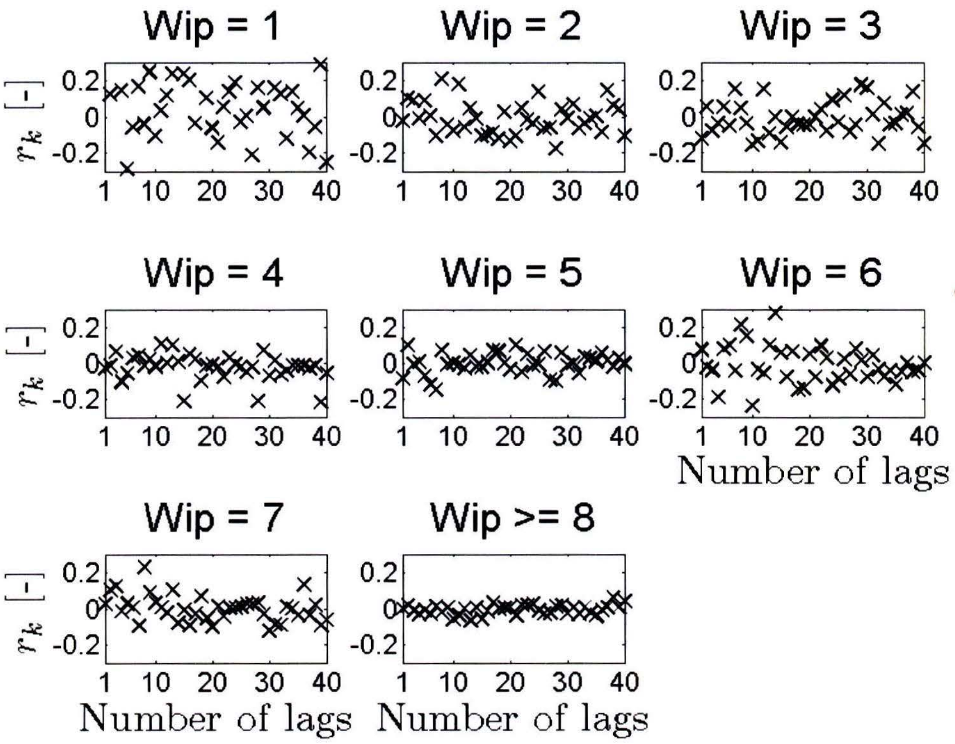


Figure 5.14: Autocorrelation factor for EPT observations.

5.3.2 Trends

The WIP dependency of the overtaking behavior is shown quite clearly in Figure 5.16. This WIP dependency, combined with the fact that the WIP level in the system was not stable during the time our data sources were created, causes a decreasing trend in the overtaking behavior. This dependency will be taken into account in the simulation model and the overtaking behavior will be analyzed per WIP level to see if there are any other trends.

Only WIP levels two up to ten are used for this, due to the low number of observations in the other buckets. The observations and a linear least squares fit are shown in Figure 5.18. There are no trends visible in the data. This suggests that the overtaking behavior should be modeled as a stationary process.

5.3.3 Autocorrelation

The last thing to check for the overtaking behavior is if there are any other dependencies within the data. The autocorrelation coefficient is again used to check this. This coefficient is shown for WIP levels two up to ten in Figure 5.19. Because of the limited number of observations the autocorrelation coefficient shows a big spread again, but it seems centered around zero. This suggests that the overtaking observations are independent of each other.

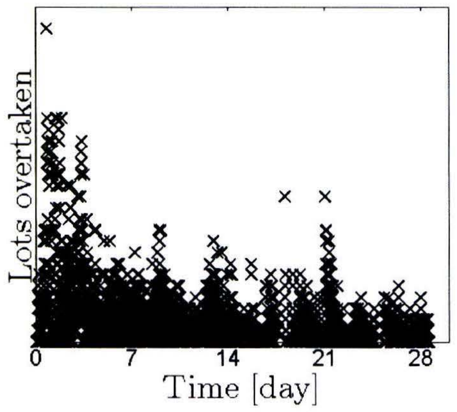


Figure 5.15: Number of lots overtaken against time.

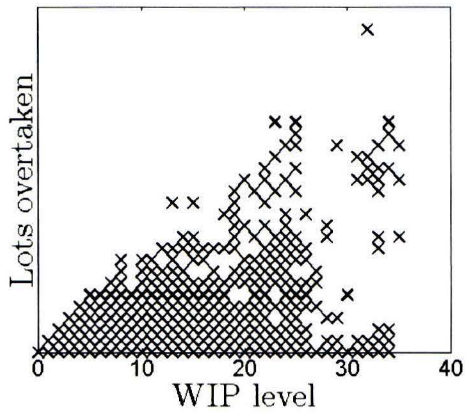


Figure 5.16: Number of lots overtaken against time.

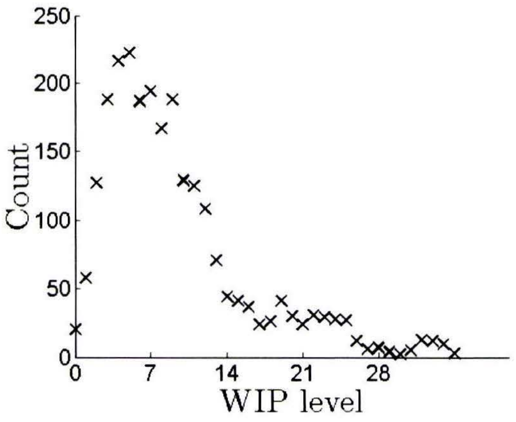


Figure 5.17: Number of observations per WIP level for the number of overtaken lots.

Based on the results in this section the overtaking behavior is modeled by independently sampling from a WIP dependent distribution.

5.4 Correlations between EPTs and inter-arrival times

A number of outliers were removed from the inter-arrival population in Section 5.1. The reasoning behind this is explained in this section.

A scatter plot of the inter-arrival times and EPT observations together is shown in Figure 5.20. These are all the observations, including all the outliers that were removed in earlier sections.

There is a clear correlation between the EPTs and inter-arrival times around days 18 and 25. From the status file it is known that the machine is unscheduled down around day 18 and the MES file shows that no new lots are arriving to the system around this time. Normal arrival

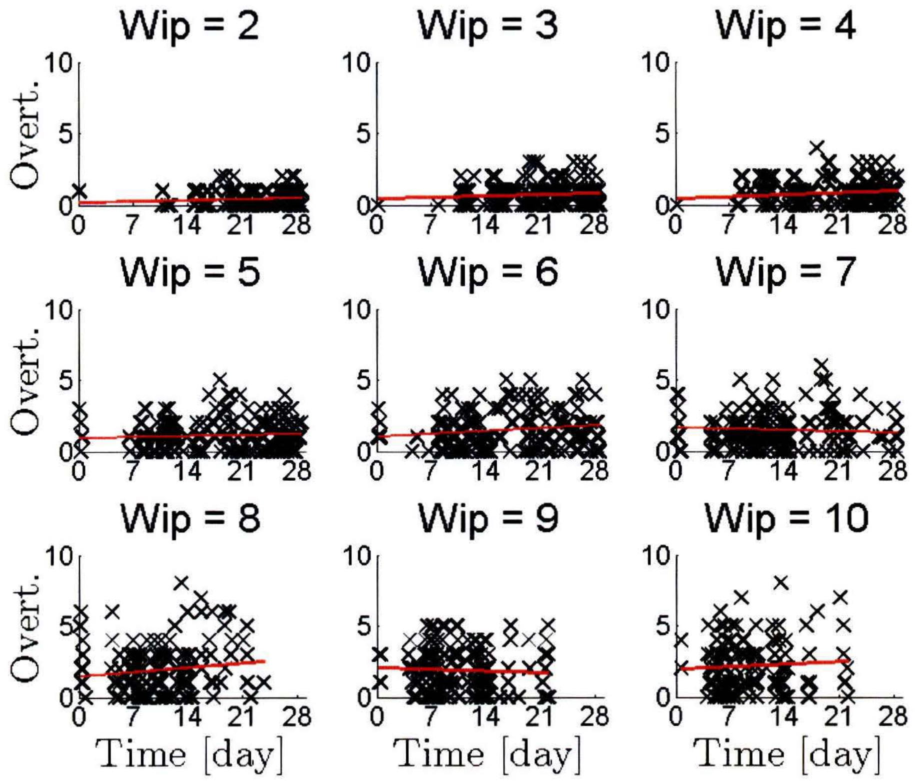


Figure 5.18: Amount of overtaken lots per WIP level, with trend lines.

resumes again after the system has been repaired. This causes the EPTs and the inter-arrival times to be high around day 18.

It is unclear what happens around day 25 but there is a short time again during which no new lots are arriving to the system. For the EPTs these outliers were identified and removed from the population. For the inter-arrival times we assume that arrivals were halted because something was wrong with the system, and choose to remove these outliers from the population.

For the rest of this thesis it is assumed that there are no further correlations between the EPTs, the overtaking behavior and the inter-arrival times. Except for the WIP dependency mentioned before.

5.5 Correlation between times to repair and time to failure

The times to failure (TTF) and times to repair (TTR) are not tested for any autocorrelation, because they are not generated by one machine. In fact they are collected from multiple machines during an entire year. Checking the autocorrelation for each machine would not give any usable results because of the small sample size. However, the number of TTF and TTR observations are the same and each TTF has a corresponding TTR observation. This

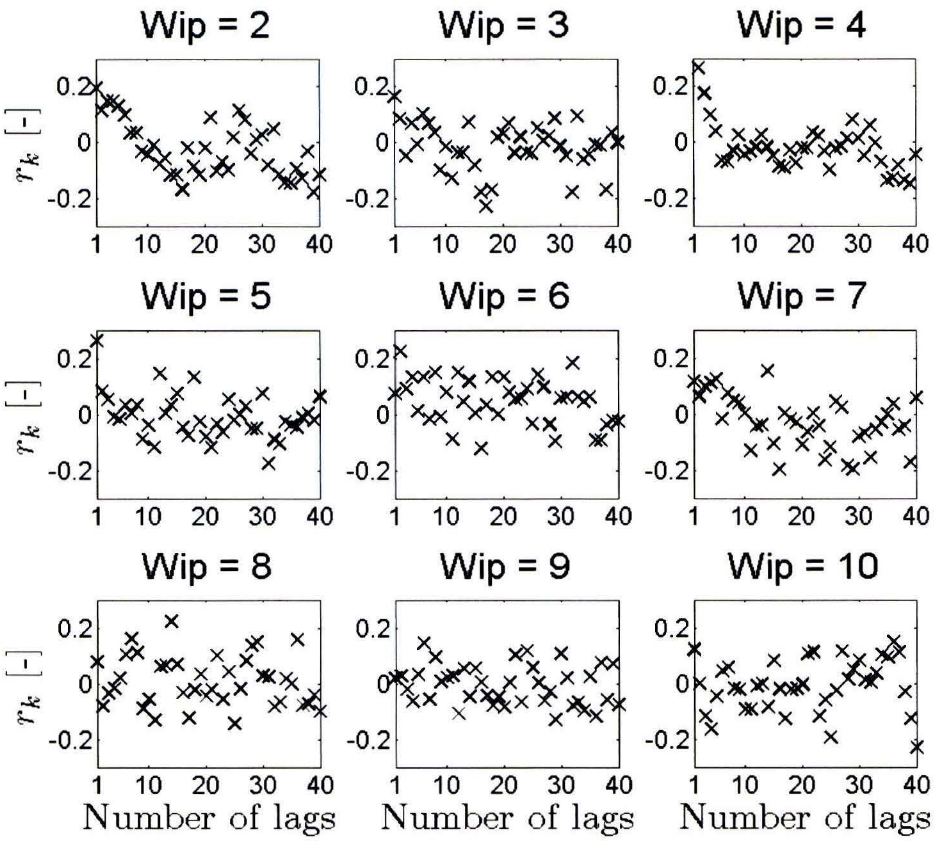


Figure 5.19: Autocorrelation factor for overtaking observations.

means that it is possible to do a quick check for any correlation between them.

Figure 5.21 shows the times to failure plotted against the corresponding times to repair. A zoom on the marked area is shown in Figure 5.22. There is no correlation visible in these plots. The correlation coefficient for the TTR and TTF is 0.035. This suggests that the times to repair and times to failure are indeed independent of each other.

5.6 Validation

After the inputs have been derived and analyzed it is possible to create a simulation model that corresponds to the actual situation. This model can be found in Appendix A. The simulation model is validated in this section. To do this, the simulated cycle times are compared with the measured cycle times.

There was one trend in the inputs that is not implemented in the simulation model: the increasing inter-arrival times. To be able to validate the simulation model the inter-arrival times are kept in the exact sequence as the measured ones. This means that the simulation run time is limited to about four weeks, the same as the original data. The times to repair and

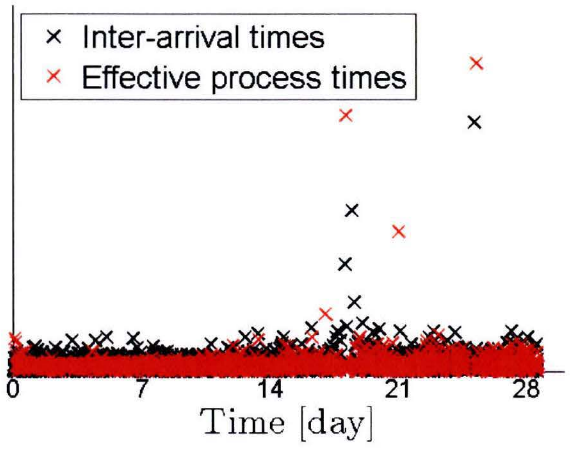


Figure 5.20: Inter-arrival times and effective process times against time.

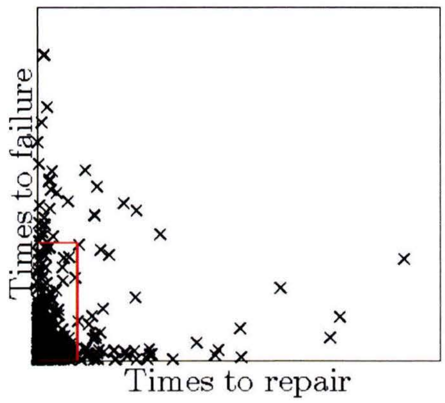


Figure 5.21: TTR against TTF.

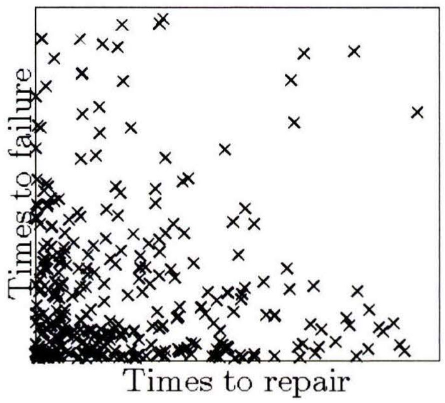


Figure 5.22: Zoom of marked area in fig. 5.21.

times to failure are also kept in the exact same sequence as the measured ones because randomizing them has to much impact on the cycle time for such a short run time. The EPT and overtaking observations for each WIP level are implemented as empirical distributions.

We are interested in the entire cycle time distribution. However, this would require plotting the distributions for all the cases that we want to run. To keep things clear and easier to illustrate we choose to focus on the mean and 90^{th} percentile of the cycle time distribution.

The central limit theorem states that the mean of a large number of random variables independently drawn from the same distribution is distributed approximately normally, irrespective of the form of the original distribution. This means that the simulated mean cycle time and the simulated 90^{th} percentile of the cycle time should follow a normal distribution if the simulation is repeated often enough. This fact is used to create confidence intervals for the simulation output.

One hundred independent simulations are performed and a 99% confidence interval is created for the mean and 90^{th} percentile values of cycle time. The mean and 90^{th} percentile value that are measured in the MES file are not included in these 99% confidence intervals. This means that the chance that we would encounter the measured mean or 90th percentile value of cycle time in our simulation results, or something more extreme, is smaller than 1%.

However, this simulation model is the only one that we have and expecting that it will produce the actual cycle times is not realistic, especially because the measured cycle time is based on one measurement of four weeks which also shows some variation. Instead, we focus on how much the simulated cycle time deviates from the measured one. Table 5.1 shows this deviation, which is defined as: $\delta = \frac{\text{simulated}-\text{meansured}}{\text{simulated}}$. The 99% confidence intervals for the simulation output are very small, the minimum and maximum values differ by less than one percent.

	δ
Mean	0.007389779
90th percentile	0.063362727

Table 5.1: Deviation between simulation and measured CT.

Although the prediction is not perfect, the mean cycle time is still very accurate with a deviation of approximately one percent. Literature suggests that anything under ten percent would indicate a good simulation model. The deviation for the 90th percentile value is also within this acceptable range. This makes us confident that the approach works and that we have a simulation model that can be used to perform a sensitivity analysis on the effect of unscheduled downs on cycle times.

Chapter 6

Distribution fitting

In the previous chapter the simulation model was validated by using empirical distributions for each of the inputs. Fitting common distributions to describe these inputs can make it easier to work with the simulation model. Furthermore, if the proper distributions are found they might describe the inputs more accurately than the empirical ones that always samples from the same, limited, set. This chapter investigates if it is possible to accurately fit distributions to our inputs. The effect of these fitted distributions on the predicted cycle times is verified at the end of this chapter.

Some of the figure axis are left blank in this chapter, due to confidentiality.

6.1 Inter-arrival times

The first input to fit a distribution on is the inter-arrival time. In the previous chapter a clear trend in the inter-arrival times was found. This means that trying to fit one distribution to the entire population will not work. However, the first 15 days of the inter-arrival data do not show this trend. The distribution fitting will be done using only these first 15 days.

Figure 6.1 shows the cumulative distribution function (CDF) for an exponential distribution with the same mean as the inter-arrival times, and the empirical CDF for the actual inter-arrival times. The corresponding probability density functions (PDFs) are shown in Figure 6.1.

The exponential distribution seems to fit the empirical ones pretty good. There are some deviations, the biggest one around the 90^{th} percentile. Formal testing of this generalization is done with the Kolmogorov-Smirnov test [13]. This test compares the maximum difference between the empirical CDF and the fitted one: $\max(|F_1(x) - F_2(x)|)$. The test returns a p value of 0.24, indicating that the probability of finding our inter-arrival times, or something more extreme, under the assumption that they are indeed from this exponential distribution, is 24%.

We chose to use this fitted distribution, despite this small chance. Mainly because this is the type of distribution that is often used in literature to model inter-arrival times.

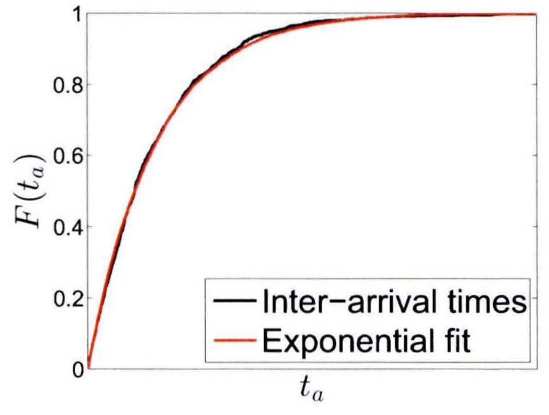


Figure 6.1: CDF for inter-arrival times.

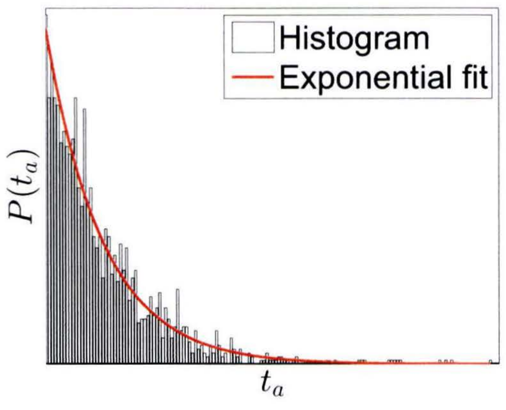


Figure 6.2: PDF for inter-arrival times.

6.2 Effective process times

The effective process times are the next input that we try to fit a distribution on. Remember that the EPT observations were divided into eight WIP buckets in the previous chapter. The empirical CDFs for each of these eight WIP levels are shown in Figure 6.3, together with a two moment fit gamma distribution. These two moment fit gamma distributions are obtained by matching the two input parameters for the gamma distributions, k and θ, to the mean and variance of the actual EPT distributions. The advantage of this method is that the fitted distribution will always have the correct mean and variance, which are the two parameters that influence the mean cycle time [8]. Other methods like maximum likelihood estimation might result in a fit that performs better in the Kolmogorov-Smirnov test, but having the correct mean and variance, and thus the correct mean cycle time, has a higher priority for us.

The corresponding PDFs are shown in Figure 6.4. There seem to be a number of small observations that cause a mismatch for the mode of the empirical distributions and the peak of the gamma distributions, the peak seems to be a bit to the left of the actual mode. Overall, the fitted gamma distribution seems to a fair choice. It seems to describe the empirical CDF fairly well for the lower WIP levels. The higher WIP levels show quite some deviation though.

Again, formal testing of these fits is done with the Kolmogorov-Smirnov test. The resulting p values for each WIP level are shown in Table 6.1. As expected, the probability that random sampling from the proposed gamma distributions would result in our actual EPT observations is very small for the higher WIP levels. WIP level one actually seems like a good fit.

Other general distributions like a log normal distribution or log logistic fit have also been fitted, but their results after being implemented in the simulation model were worse. The same goes for fitting the EPT distributions with an offset.

We could try a different distribution for each WIP level, but this seems undesirable as previous research in semiconductor EPT modeling have used gamma distributions with great success. So even though the formal test rejects the hypothesis that the two moment fit gamma distri-

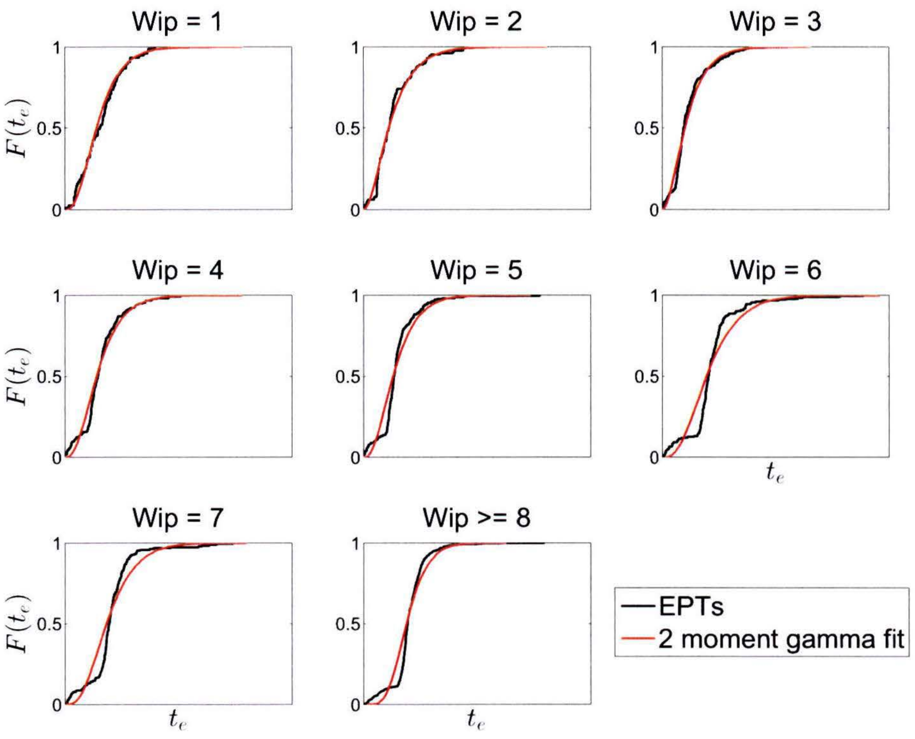


Figure 6.3: CDF for EPTS and two moment gamma fits.

butions are good generalizations, it will still be tested in the simulation model and compared to using the empirical EPT distributions in Section 6.6.

6.3 Overtaking behavior

Literature did not provide us with any distribution to fit the number of overtaken lots. An attempt was made at parameterizing the overtaking behavior in Chapter 7. Unfortunately this attempt was not successful. The overtaking behavior will thus be described by empirical distributions.

6.4 Times to repair

Several distributions are suggested in literature to model times to repair. The most common one that applies to our system is the gamma distribution. The empirical CDF and the CDF for the two moment fit gamma distribution are shown in Figure 6.5. The corresponding PDFs are shown in Figure 6.6. Graphical inspection already reveals that there is quite some deviation from the 80^{th} percentile upwards. The Kolmogorov-Smirnov test indicates that the

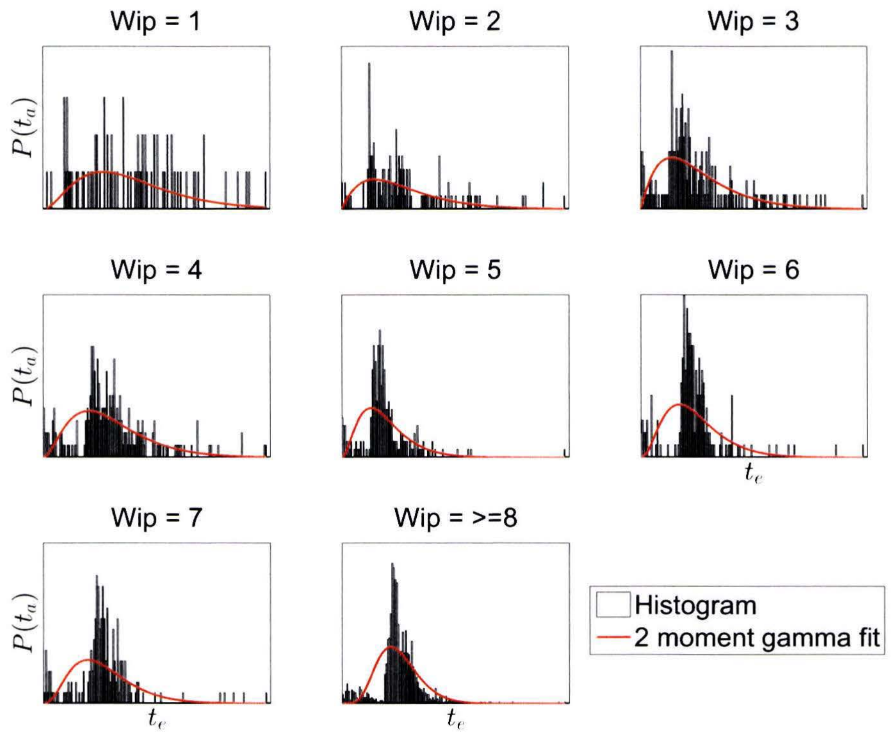


Figure 6.4: PDF for EPTS and two moment gamma fits.

maximum deviation between the two CDF functions is too big with a resulting p value of 1.2407e-15.

The time to repair is influenced by several policies, like the amount of training of the local staff and the policy for stocking spare parts. This is apparently not described accurately by a gamma distribution. However, it will still be tested with the simulation model and the accuracy will be compared to using the empirical distribution to see how much it affects the prediction accuracy.

6.5 Times to failure

The final input to fit a distribution to is the time to failure. Figure 6.7 shows the empirical CDF for the times to failure, and the CDF for the two moment fit gamma distribution. The corresponding PDFs are shown in Figure 6.8. The two CDFs seem pretty similar, except that the empirical one shows more noise. The Kolmogorov-Smirnov test gives a p value of 0.00036711, making it easy to reject the gamma distribution as an accurate fit. The gamma distribution is still tested with our simulation model to see how much it influences the accuracy. From literature we would expect the time to failure to be exponentially distributed. An exponential fit with the same mean was also tested but had significantly worse results, both before and after reconstruction in Section 4.3.1. This can indicate that the time to

WIP level	P value
1	0.5804
2	0.02759
3	0.013001
4	0.0017463
5	5.0408e-08
6	1.1874e-09
7	6.3742e-08
8 or more	5.9149e-34

Table 6.1: KS goodness of fit test for two moment fit gamma distributions.

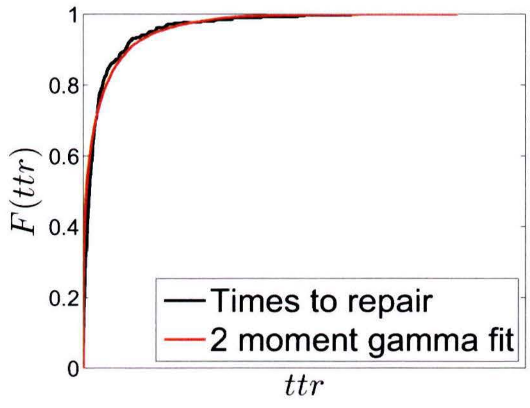


Figure 6.5: CDF for times to repair.

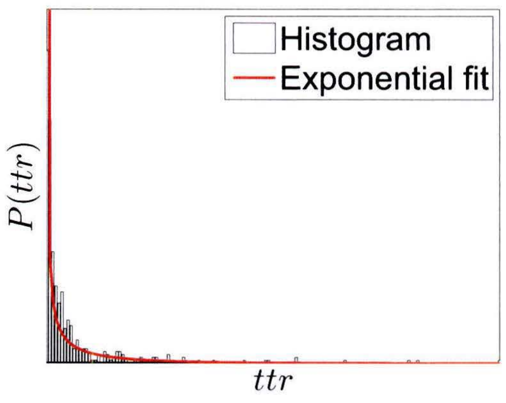


Figure 6.6: PDF for times to repair.

failure is actually not memoryless and failures can be predicted to some extend. However, this is beyond the scope of this thesis.

6.6 Verification of the fitted distributions

The simulation model takes five inputs: The inter-arrival times, the effective processing times, the overtaking behavior, the times to repair and the times to failure. The only input that was described fairly successful with a fitted distribution are the inter-arrival times. It was not possible to fit the overtaking behavior at all. For all other inputs some fitted distributions were proposed that describe them to some extend and we are interested to see how they influence the simulation output.

This test is a little different from the validation in Section 5.6. For this test the exponential fit on the inter-arrival times will be used, instead of the actual, time dependent inter-arrival stream in the validation. Furthermore, the failure behavior is now based on the behavior of many machines during one year, instead of the actual stream from the Status file. This means that it is no longer possible to compare the values that the simulation produces to the measured ones, because it is simulating at another utilization level. However, this is not a problem, since the simulation model has already been validated in Section 5.6.

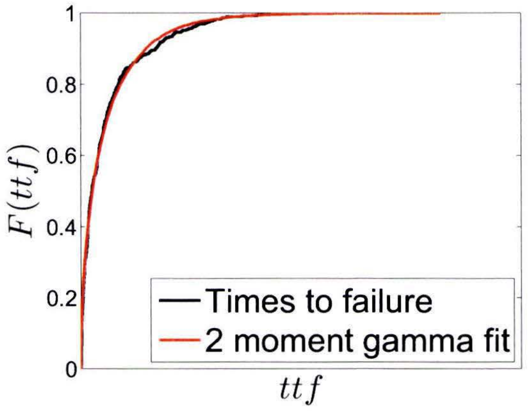


Figure 6.7: CDF for times to failure.

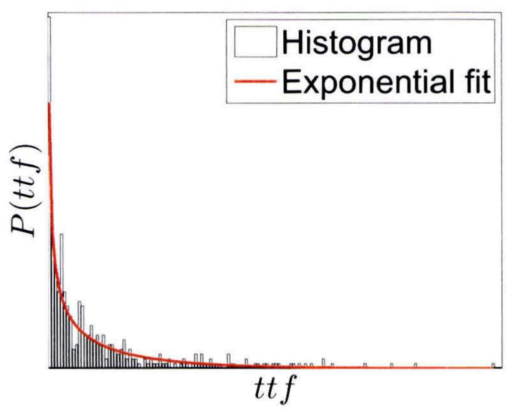


Figure 6.8: PDF for times to failure.

Before the different fitted distributions are compared, the time for the system to get in a steady state is investigated. Case 0 from Table 6.2 is simulated three times, simulating nine years every time. The resulting cumulative moving averages of the cycle time is shown in Figure 6.9.

It can be seen that for all three runs the cumulative moving average of the cycle time seems to stabilize after approximately five simulated years. This is much longer than the slowest process, the time to failure, so any more jumps in the cumulative moving average if the simulation time is increased are not expected. We assume that after five simulated years the 90^{th} percentile value has also stabilized. Furthermore we assume that the fitted distributions will not affect this much. Based on this, a run length of five years is used for the following simulations.

Comparing the fitted distributions with the empirical ones, for the EPTs, times to repair and times to failure results in 2^3 different possible combinations of inputs for the simulation model. These combinations are shown in Table 6.2. Case 0 is the first case that is simulated. This will provide a baseline value to compare the effect of the different distributions. Every case in Table 6.2 is simulated a total of ten times. The resulting outputs all have a confidence interval of which the maximum and minimum deviate less than 0.5% from the mean. This interval is so small that only the mean values are used in the comparison. The deviations, compared to the baseline, of the simulated mean cycle time and 90^{th} percentile are shown in the last two columns for each run. This deviation is defined as $\delta = \frac{\text{simulated}-\text{baseline}}{\text{simulated}}$.

It can be seen that using the gamma fits for the effective process times causes quite a big deviation of the simulated cycle times. Any run where the empirical EPT distributions are replaced with the fitted gamma distributions show a big deviation from the baseline. Using the fitted distributions for the times to repair and times to failure seems to have little effect on the cycle times and 90^{th} percentile values.

Based on this, the gamma fits for the times to repair and times to failure will be used in the rest of the thesis. The empirical distributions will be used for the effective process times. Using the empirical EPT distributions is not a problem because, opposed to the time to repair and time to failure, we do not plan to study any changes to them.

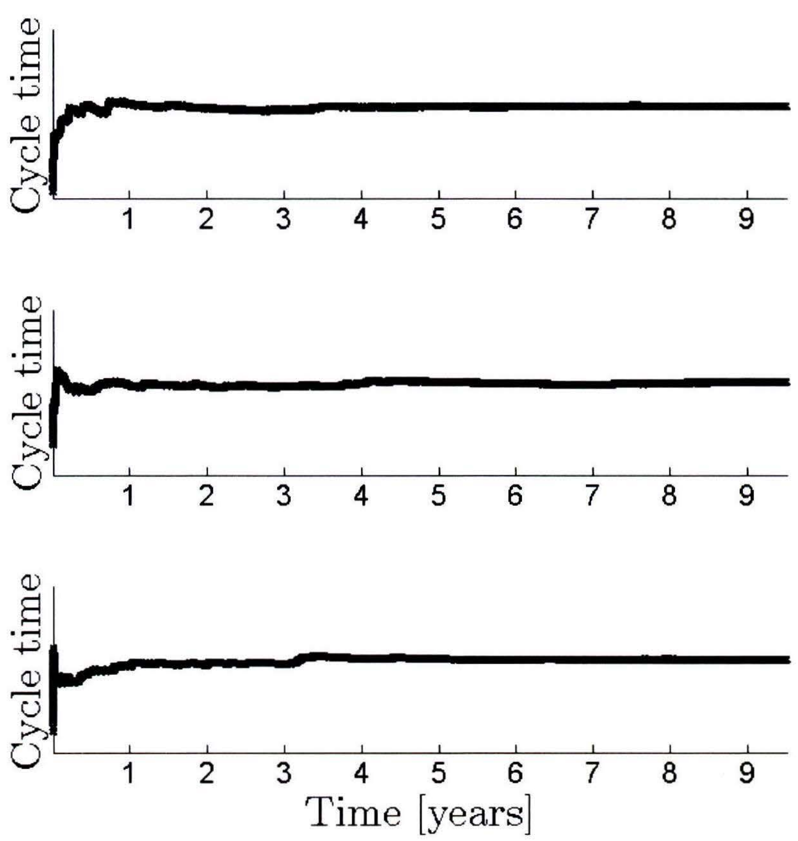


Figure 6.9: Cumulative moving average cycle time, over time.

Case	Arrivals	EPT	Overtaking	TTR	TTF	δ mean	δ $90^{th}p$
0	Exponential	Empirical	Empirical	Empirical	Empirical	-	-
1	Exponential	Gamma	Empirical	Empirical	Empirical	0.180	0.220
2	Exponential	Empirical	Empirical	Gamma	Empirical	0.003	0.003
3	Exponential	Empirical	Empirical	Empirical	Gamma	0.008	0.012
4	Exponential	Gamma	Empirical	Gamma	Empirical	0.187	0.231
5	Exponential	Gamma	Empirical	Empirical	Gamma	0.199	0.249
6	Exponential	Empirical	Empirical	Gamma	Gamma	0.011	0.019
7	Exponential	Gamma	Empirical	Gamma	Gamma	0.195	0.246

Table 6.2: Validation runs with results.

Chapter 7

Reconstructing lot priorities

It is common practice in manufacturing to use a FIFO policy for their buffers, but to give some lots high priority. Lots with high priority will overtake lots with low priority. In this chapter we attempt to reconstruct the priorities that were assigned to different lots, based on the arrival and departure times to the buffer. Benefits of this approach are that the overtaking behavior might no longer be WIP dependent and further studies can be conducted to the effect of changing lot priorities.

7.1 Concept

The approach that we propose uses the arrival and departure times to the buffer for each lot. With this data it should be possible to set up a set of constraints to describe the relative priorities of the lots. This approach is explained in the following example. Assume the following arrival and departure sequence for a given set of five lots, and the corresponding queue composition:

Time	Event	LotID	Queue
1304.525	Arrival	0	[0]
1740.244	Arrival	1	[0,1]
2290.986	Arrival	2	[0,1,2]
2525.514	Arrival	3	[0,1,2,3]
3433.041	Departure	1	[0,2,3]
3181.931	Arrival	4	[0,2,3,4]
4486.872	Departure	0	[2,3,4]
5158.733	Departure	4	[2,3]
5427.477	Departure	2	[3]
6099.338	Departure	3	[]

Table 7.1: Arrival and departure data from the buffer.

It can be seen that lot one has a higher priority than lot zero, because it arrives in the queue later but it leaves earlier. The same goes for lot four which arrives in the queue after lots two and three, but leaves before them. Furthermore it can be seen that priorities for lots two and three have to be lower or equal than the priority of lot one, because they are in the queue at the time that lot one leaves and do not overtake it.

The example from table 7.1 can be rewritten into a set of equations by using this logic. In order to do this some assumptions are made:

1. All priorities are integers, described by p_i.
2. The higher the priority, the higher the value of p_i.

This leads to the following set of equations that constraint the lot priorities:

$$\begin{aligned}
p0 &\leq p1 - 1\\
p2 &\leq p4 - 1\\
p3 &\leq p4 - 1\\
p2 &\leq p1\\
p3 &\leq p1\\
p2 &\leq p0\\
p3 &\leq p0\\
p4 &\leq p0\\
p3 &\leq p2
\end{aligned}$$

This can be rewritten into matrix form:

$$\begin{bmatrix} 1 & -1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & -1\\ 0 & 0 & 0 & 1 & -1\\ 0 & -1 & 1 & 0 & 0\\ 0 & -1 & 0 & 1 & 0\\ -1 & 0 & 1 & 0 & 0\\ -1 & 0 & 0 & 1 & 0\\ -1 & 0 & 0 & 0 & 1\\ 0 & 0 & -1 & 1 & 0 \end{bmatrix} \begin{bmatrix} p0\\ p1\\ p2\\ p3\\ p4 \end{bmatrix} \leq \begin{bmatrix} -1\\ -1\\ -1\\ 0\\ 0\\ 0\\ 0\\ 0\\ 0 \end{bmatrix}$$

The objective is then to find the lowest possible priorities for each lot, that satisfies these constraints. This can be solved using integer programming. The objective function is to minimize $\sum_{i=1}^{4} p_i$. This objective function is linear for all parameters and adding a constraint that all priorities have to be greater of equal to zero will results a minimum.

For this simple case it can be seen that the best solution is:

$$
\begin{aligned}
p0 &= 1\\
p1 &= 2\\
p2 &= 0\\
p3 &= 0\\
p4 &= 1
\end{aligned}
$$

7.2 Test case

This approach is tested using simulation in χ [22]. A single machine with an infinite buffer is simulated. A generator produces lots, each lot has a certain priority which is drawn from a uniform distribution with possibilities zero, one and two. A higher number means a higher priority. After they are generated, lots are inserted into the queue based on their priority number. They are inserted before any lots with a lower priority, and after any lots that have the same or a higher priority. The machine takes the first lot in the queue as soon as it can process a new lot. This means that besides the priority system, it works according to a FIFO policy.

The method described in the previous section is implemented in Matlab and the results are shown in Figure 7.1. The real priority from the χ simulation is saved and is called p. The reconstructed priority is called $\hat{p}$. It can be seen that the reconstructed priority tends to underestimate the real priority. The reconstructed priority is only correct if lots with all possible priorities are present in the calculation for each lot. In all other cases the reconstructed priority is too low.

In our test case for example there could be another set of lots that start arriving after lot three has left the system. Imagine that all the lots in this second set have priority three. Calculating the priorities for this set would result in all these lots having a reconstructed priority of one.

Between lot 1500 and 1600 the reconstructed priorities are correct. Using only this part of the data to recreate the priority distribution would yield an accurate result. Unfortunately, in the real data, there is no way of knowing which part of the reconstructed priorities is correct and which is not.

Figures 7.2 and 7.3 show the distributions for the real and the reconstructed priorities. The reconstructed priorities show a different distribution than the actual one. The number of lots with priority zero is estimated almost correctly, while the number of lots with priority one is overestimated and the number of lots with priority two is underestimated. We have to conclude that this approach does not work for reconstructing lot priorities. The empirical overtaking behavior distributions, calculated using the algorithm in Appendix D, will be used in the simulation model.

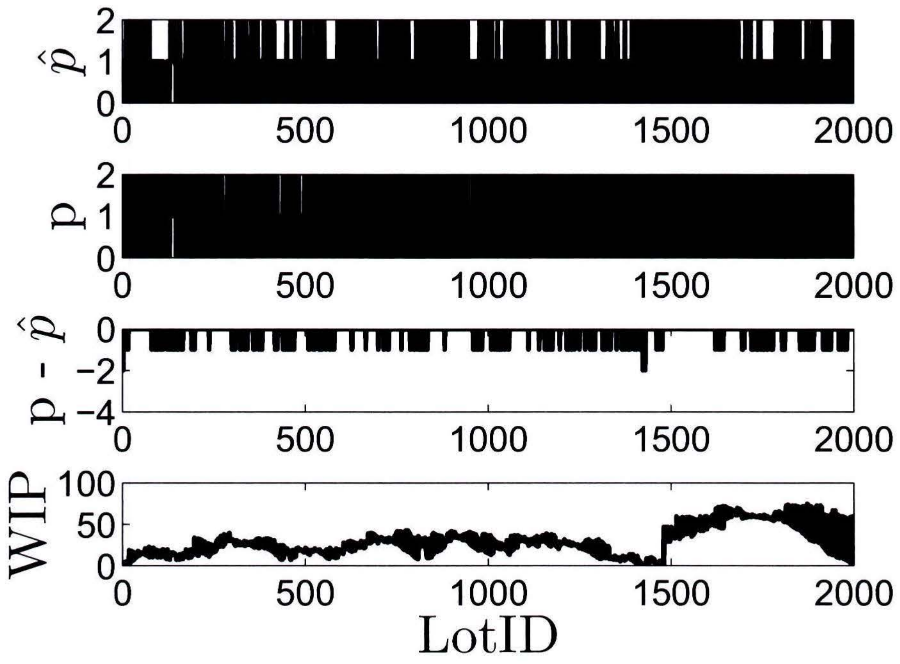


Figure 7.1: Test case using χ simulation.

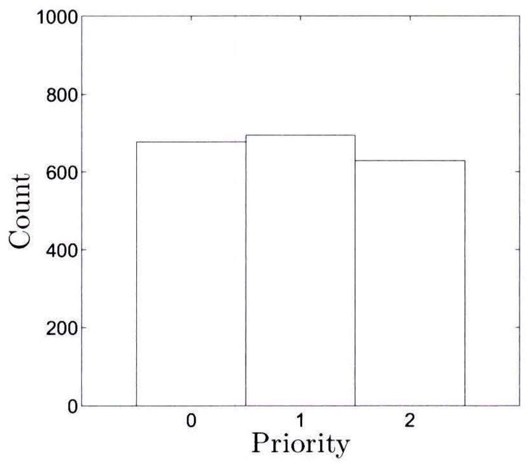


Figure 7.2: Original priority distribution.

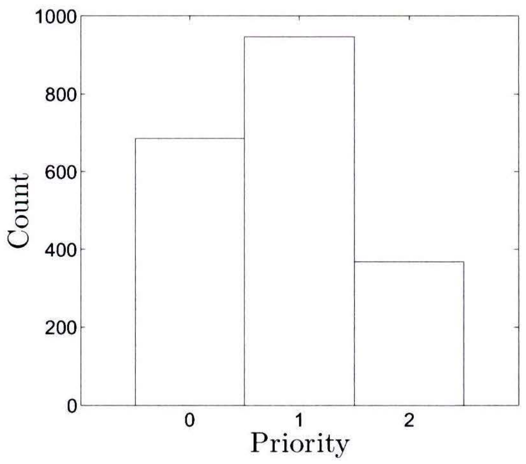


Figure 7.3: Reconstructed distribution.

Chapter 8

The effect of unscheduled downs

After the simulation model has been created and validated in the previous chapters, it is time to perform the sensitivity analysis to find the effect of unscheduled downs on cycle times. This chapter will study changes to five parameters of the unscheduled down behavior: the mean time to repair, the coefficient of variation of the time to repair, the mean time to failure, the coefficient of variation of the time to failure and removing the top five percentile of the times to repair. The effect of these five parameters is shown in the following sections.

The simulations are ran at an utilization level of approximately 90%. This is a common utilization level based on Figure 4.5. Lowering the utilization level means that the effects on cycle time that are found in this chapter will also decrease [15]. All of the cycle time plots in this chapter are normalized due to confidentiality.

8.1 Queueing theory

Queueing theory will be used to validate the trends that are found in the following sections. The closest analytical estimation that is available is derived for a M/G/1 system with time dependent failures [3], where the time to failure is exponentially distributed and the time to repair is generally distributed:

$$E(S) = \frac{\rho_G E(R_G)}{1-\rho_G} + \frac{E(D)\eta}{1+E(D)\eta} E(R_D) + E(B) + E(B)\eta E(D)$$

With:

$$\begin{aligned} \rho_g =& \quad \lambda(E(B) + E(B)\eta E(D)) \\ E(R_G) =& \quad \tfrac{E(B^2)+(1+\eta E(D))^2+E(B)\eta E(D^2)}{2(E(B)+E(B)\eta E(D))} \\ E(R_D) =& \quad \tfrac{E(D^2)}{2E(D)} \end{aligned}$$

Here $E(S)$ is the mean cycle time, $E(B)$ is the mean processing time and $E(B^2)$ is the variance of the processing times. Similarly $E(D)$ and $E(D^2)$ are the first and second moment of the times to repair. $\frac{1}{\eta}$ is the average time to failure.

Four of the five cases that are investigated in the following sections can also be investigated using this equation. However, there are some important deviations between our model, and an M/G/1 system:

1. While the simulation model does have general processing times, they are dependent on the WIP level in the system.
2. The time to failure in the simulation model is gamma distributed, instead of exponentially.

The fact that the processing times for the simulation model decrease when the WIP level increases, creates a sort of self correcting mechanism. For example: If a machine experiences longer repair times, the WIP level will increase. This will in turn reduce the processing time. This dampens the effect of the longer repair times in the simulation model, compared to the analytical M/G/1 model. This means that the queueing model can not be used to quantify the cycle times for these cases, but it can still be used to find whether a change should increase of decrease cycle times.

8.2 Mean time to repair

The first case studies the effect of changing the mean time to repair. This is done by reducing the scale parameter of the gamma distribution that was fitted to the time to repair. This will change the mean of the distribution, while keeping the coefficient of variation the same.

Some suggestions to reduce the time to repair on an actual machine are to store spare parts closer to the actual machine or train the local teams to solve problems faster. The mean time to repair is varied from 90% to 110% of the original value. The 95% confidence intervals on the mean and 90th percentile of the simulated cycle times are shown in Figures 8.1 and 8.2.

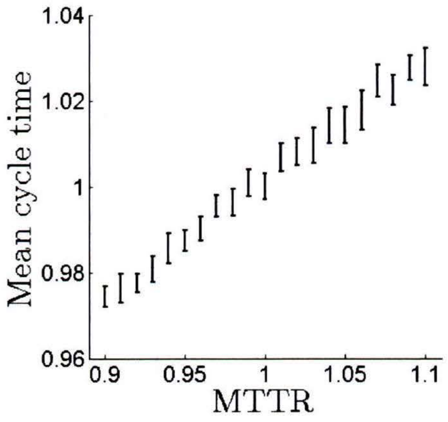


Figure 8.1: Mean CT vs. MTTR.

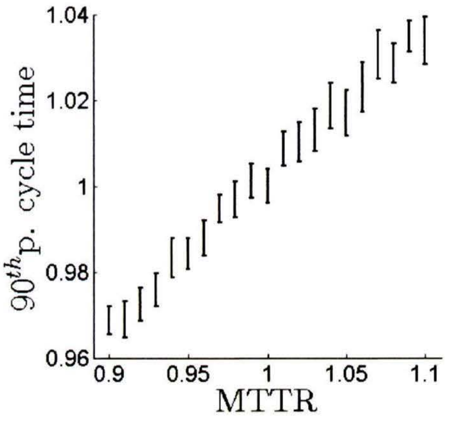


Figure 8.2: 90^{th} percentile CT vs. MTTR.

It can be seen that reducing the mean time to repair by 10% yields a reduction of the mean and 90th percentile of the cycle times of approximately 2.5%. Queueing theory also indicates that reducing the mean time to repair should result in smaller cycle times. This is because shorter repair times lead to smaller outliers in the effective processing times, which means means a smaller mean and variance of the effective processing times.

8.3 COV time to repair

The second case studies the effect of changing the variance of the time to repair, while keeping the mean the same. This results in changing the coefficient of variation for the time to repair (c_r). It is done by changing the shape parameter of the gamma distribution, and adjusting the scale parameter to keep the mean the same. Practical ways to reduce the variance could be reduced stocking of spare parts that cause short machine downs, in favor of spare parts that cause very long down times. The coefficient of variation is varied from 90% to 110% of the original value. The resulting 95% confidence intervals for the mean and 90th percentile of the cycle time are shown in Figures 8.3 and 8.4.

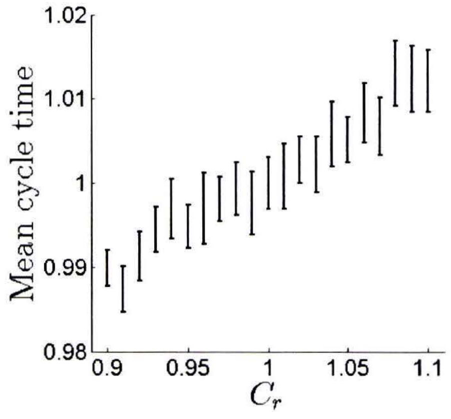


Figure 8.3: Mean CT vs. c_r.

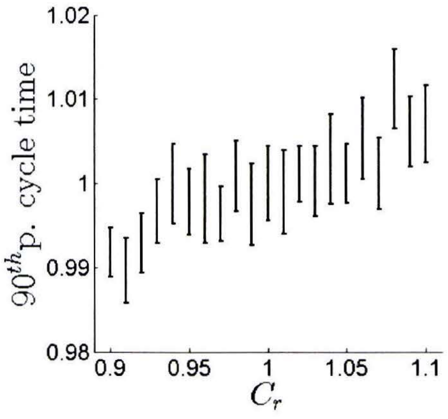


Figure 8.4: 90th percentile CT vs. c_r.

It can be seen that reducing the coefficient of variation of the times to repair decreases the mean and 90th percentile of the cycle times. A ten percent decrease in c_r leads to an approximately 1% lower mean and 90th percentile of cycle time. Queueing theory indicates the same trend: decreasing cycle times for decreased variance of times to repair. The reason behind this is that less outliers in the times to repair means that there are less big outliers in the effective processing times.

8.4 Mean time to failure

The third case studies the effect of changing the mean times to failure, while keeping the coefficient of variation the same. The approach is the same as the first case where the mean time to repair was changed. Increasing the time to failure can be achieved in reality by making the machine parts more durable. A litho scanner consists of millions of parts, all of which can break and stop the machine from producing. Making all of them more durable is a very big challenge. Nevertheless, we still investigate the effect on cycle time. The results are shown in Figures 8.5 and 8.6.

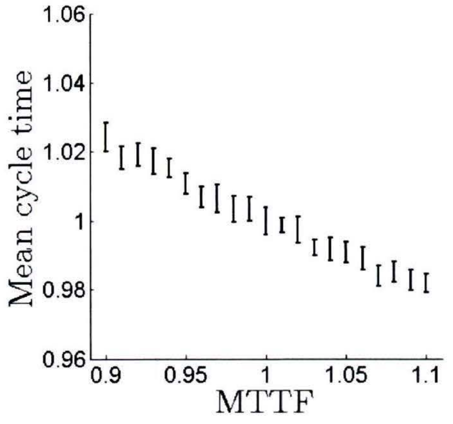


Figure 8.5: Mean CT vs. MTTF.

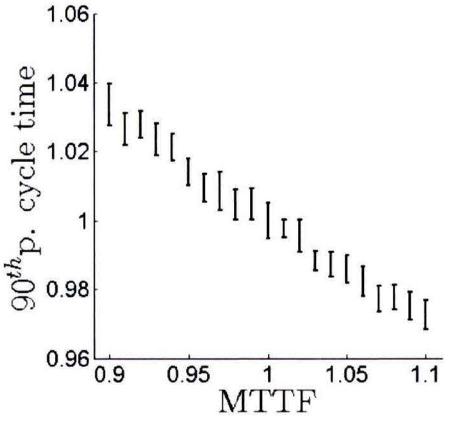


Figure 8.6: 90^{th} percentile CT vs. MTTF.

It can be seen that increasing the mean time to failure by 10% will decrease the mean cycle time by approximately 2%, while decreasing the $90th$ percentile cycle time by about 3%. Queueing theory also predicts this same trend. Increasing the mean time to failure means there are less failures which reduces both the mean and variance of the effective process times.

8.5 COV time to failure

The fourth case studies the effect of changing the coefficient of variation of the time to failure (c_f), while keeping the mean the same. Reducing the COV of the time to failure can be achieved by focussing the design on durability for parts that tend to break often, while focussing on weight reduction or other improvements for parts that do not break often.

It can be seen that reducing the coefficient of variation of the times to failure has a very small effect on the cycle times. A 10% decrease in the COV leads to a decrease of approximately 0.3% for the mean and 90^{th} percentile of the cycle time. This parameter is not present in the queueing theory approximation, but the trend in these results still seems right. Decreasing the coefficient of variation of the times to failure means the failures will be spread more evenly over time. This results in less failures shortly after each other, which ccould cause WIP to pile up.

8.6 Removing long downs

The fifth case studies the effect of removing all observations that are in the $(100 - x)^{th}$ percentile of the original times to repair, and use the resulting population as an empirical

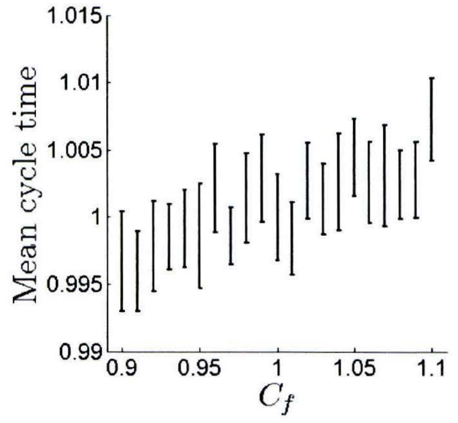


Figure 8.7: Mean CT vs. c_f.

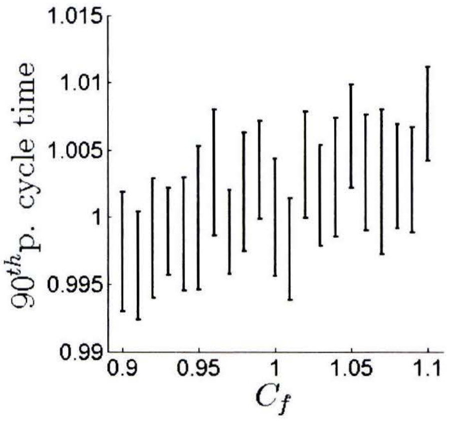


Figure 8.8: 90^{th} percentile CT vs. c_f.

distribution. This could be achieved by storing more spare parts on-site, training the local team to be able to solve more complex problems, or making parts that cause problems because they are difficult to reach, more accessible. The results are shown in Figures 8.9 and 8.10.

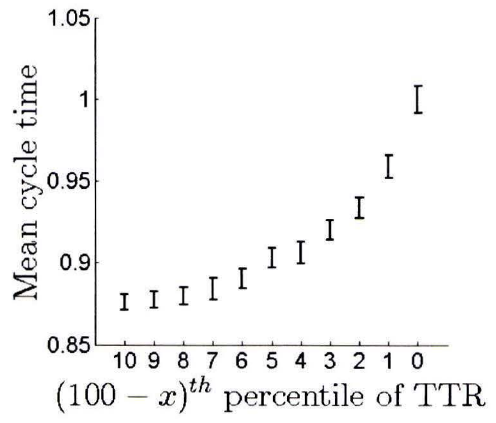


Figure 8.9: Mean CT vs. TTR.

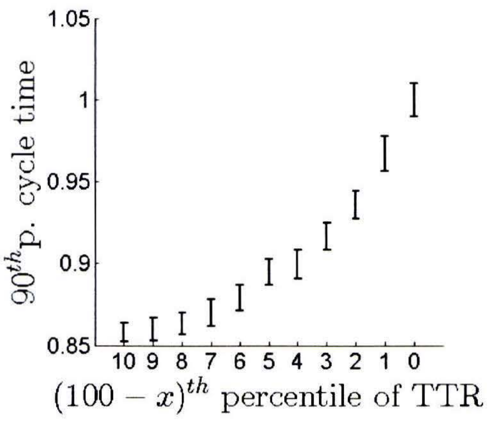


Figure 8.10: 90^{th} percentile CT vs. TTR.

It can be seen that removing long downs has a very big effect on the cycle time. Removing the biggest 10% of the repair times yields a decrease of approximately 12% for the mean and 14% for the 90^{th} percentile of the cycle times. This is because the outliers of the times to repair are removed, combining the effects of Sections 8.2 and 8.3. The results here are more prominent because removing the top tenth percentile values will reduce the mean and COV of the time to repair by 50 and 40%, respectively.

Chapter 9

The effect of matching machines

This chapter investigates the effect on cycle time of matching two machines. Each scanner has it's own specific imperfections which lead to specific offsets in the exposed layers. These offsets are called the machine signature or machine fingerprint. Matching machines is a method to make two machines mimic each others so called fingerprints.

In semiconductor manufacturing the relative positioning of two consecutive layers sometimes requires a precision that is on the very edge of the scanners capabilities. These layers are called critical layers. Common practice is to create these critical layers on the same machine, to avoid extra inaccuracy that another machine fingerprint will add. This practise is called machine dedication.

Machine dedication has one big disadvantage: In case of a machine failure, all dedicated WIP will have to wait until the machine is fixed before it can be processed again. This can lead to long and unpredictable cycle times. Sometimes cycle times get so out of control that dedicated lots are processed on another machine, and the resulting loss of quality is accepted.

There is an alternative way to solve this problem. It is possible to reprogram a number of machines, so they copy each others fingerprint on top of their own. This method requires a lot of time to recalibrate the machines, which has to be repeated every time an important part of one of the machines breaks and gets replaced. This makes it impractical to match more than two machines together. The full process of matching is complex [17] and we are not going into more details here.

The process of matching is of course not perfect, but it can reduce the number of layers that require machine dedication. The benefit is that as the number of dedicated layers is reduces, the system will start to act more and more like two parallel machines, instead of single machines. This means that if one machine breaks down, the other can take over part of the workload. This reduces the effect of the machine downtime, and other hiccups, on cycle time.

The possible benefit of matching two machines is investigated by simulating two parallel machines. A percentage of the products that arrive to these machines are dedicated to one of the two machines, while the rest can be processed by either machine. The amount of dedicated lots is ranged from 0 to 100%, where 100 means every arriving lot is dedicated, and

0% means that every arriving lot can be processed on either machine.

9.1 Model extensions

The model that was introduced in Chapter 3 is used as a basis to create a model to analyse the effect of matching two machines. The extended model is shown in Figure 9.1.

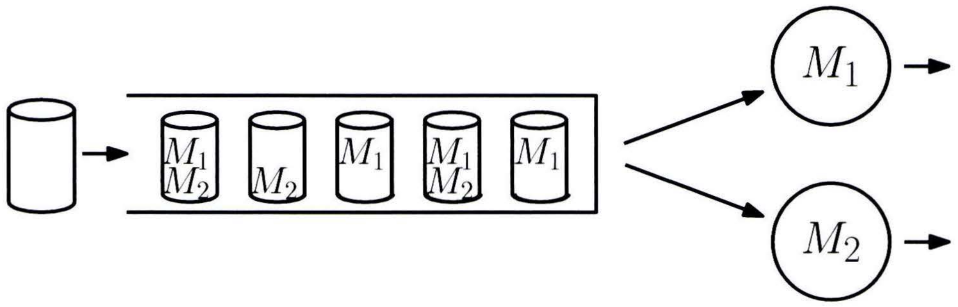


Figure 9.1: Extended model for machine matching.

The first change to the previous model is the number of machines. In the extended model there are two machine, instead of one. Upstream of these two machines is one infinite buffer to which all lots arrive. A machine will take the first lot that it can process from this buffer, as soon as it is ready to process a new one. The percentage of dedicated lots that arrive to the buffer is varied. Half of all dedicated lots are assigned to the first machine, and the other half to the second machine. In reality lots might be dedicated to the machine that their first critical layer was processed on. This means that after one machine was down, there might be an increase in the number of dedicated lots for the other machine. This effect is not taken into account in our modeling approach.

Both of these machines are subject to time dependent failures. A machine is preemptively interrupted once an unscheduled down occurs. After the repair time has passed, the machine will continue to work on the lot at the point where it was interrupted. There is no scrappage of lots, similar to the original model. The failures of these two machines are independent of each other, but follow the same distributions. Furthermore, each machine has WIP dependent effective processing times. These processing times are sampled independently of each other. The WIP for one machine is determined by counting the number of lots that are dedicated to that machine, and adding half of the lots that can be processed on either machine to this number. This assumes that each machine will process about half of the lots that are not dedicated. During a long downtime for one machine this might cause an inaccurate estimation of the WIP level for the other machine, because all WIP will be processed on the second machine until the first one is repaired. However, a long machine down for one machine means that the WIP level will be high, and a high WIP level means that the EPTs are already at their minimum.

The fitted distributions for the times to repair and times to failure are used in the model. The fitted distribution for the inter-arrival times is also used, but the average inter-arrival time is halved to compensate for the fact that there are two machines now. The EPT observations are sampled from an empirical distribution again. The overtaking behavior is not taken into account in this chapter. The only overtaking distribution that is available was measured for one machine. It is unclear how this behavior is affected if two machines with different qualifications are used in parallel. Furthermore the observations from our data are measured for a range of WIP levels that occur for one machine. If two machines operate in parallel, the total WIP level will be up to two times as high. This makes the range of WIP levels in our original observations too small to use them. Not taking the overtaking behavior into account means that the model can only predict the mean cycle time, instead of the cycle time distribution.

The EPT observations are also WIP dependent. However, there was a clear lower limit that was reached for a WIP level of 8 or higher. This means that it is still possible to use the empirical, WIP dependent, EPT distribution. Even if the WIP level would double.

This model is implemented as a discrete event simulation model. The details of this model can be found in Appendix B.

9.2 Effect of matching

The model from the previous section is used to measure the effect of matching two machines to each other. Matching machines means that fewer lots have to be dedicated to one machine, and can instead be processed on either machine. The percentage of arriving lots that are dedicated is varied from 0 to 100%. The simulated 95% CI for the corresponding mean cycle times are shown in Figure 9.2. Note that the cycle times are normalized due to confidentiality. These results are simulated at a machine utilization level of approximately 90%, which is a common utilization level according to Figure 4.5. It can be seen that the cycle time in the system is lowered as the number of dedicated lots is decreased. A system without dedicated lots has almost half the cycle time time of a system where all lots are dedicated. Furthermore, it can be seen that decreasing the number of dedicated lots from 100% to 75% will already reduce cycle times by 25%.

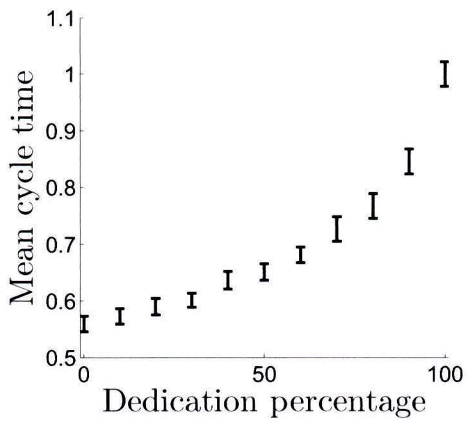


Figure 9.2: Cycle time for dedication levels.

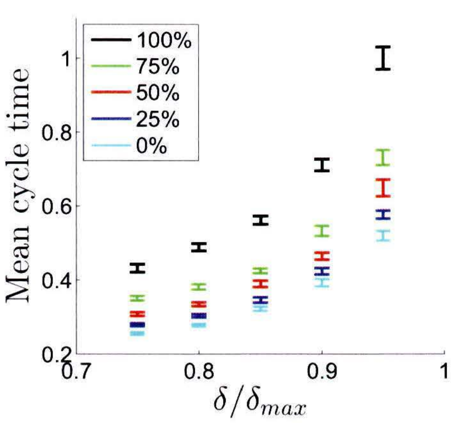


Figure 9.3: CT-TP curves for dedication levels.

The same simulations are also performed for different utilization levels. This provides us with the Cycle time - Throughput (CT-TH) curves for different levels of lot dedication. The resulting 95% CI are shown in Figure 9.3. It can be seen that the gain in cycle times decreases as the utilization level is lowered. The utilization level in these plots is defined as the actual throughput, δ, divided by the maximum achievable throughput for this system, δ_{max}.

Chapter 10

Detailed model

The thesis so far has focussed on using a full aggregate model to find the effect of unscheduled machine downs and machine matching, on cycle times. While these two effect have a big impact, they also require big effort to change. For a system running at a customer site there might also be other possibilities to improve cycle times, that cost less to implement.

This chapter introduces a more detailed model, that divides the litho cell into three parts:

1. Dispatching delay and transportation

2. Loading

3. Processing

The first part is called *Dispatching delay and transportation.* This is the time it takes before a lot arrives on the litho cell from the stocker, after a port has become free. This time includes everything that happens in between such as transportation of the wafer from the stocker to the litho cell or waiting for the lot to be inspected by an operator. The second part, *Loading*, is the time for all the wafers to be loaded from the FOUP into the litho cell. This includes opening the FOUP and loading all the wafers from the FOUP into the track. The third part is called *Processing.* This contains everything that happens after the wafers are loaded into the litho cell, until they leave the system.

Dividing the litho cell into three parts makes it possible to study more what-if scenarios. For instance: What happens if the loading times are reduced? With the full aggregate model this would require an estimation of how this impacts the EPT observations (and changing them accordingly), or adjusting the MES file to change all the departure times and recalculating the EPT observations. With this new model such changes are easier to predict as the EPTs, for instance for the loading part, can be changed directly. Another benefit is that as there are three sets of EPT observations now, it is immediately clear which part of the litho cell has the most outliers, or other irregularities.

10.1 Model

Dividing the litho cell into three parts requires a new model to analyse the system. This new model is obtained by linking the three parts of the system together. This is done based on two known capacity restrictions in the litho cell: The number of ports, which is the maximum number of lots that can be on the litho cell simultaneously, and the restriction that only one lot can be loaded into the litho cell at a time. For the system in our data the number of ports is equal to five. The resulting new system is called the detailed model and it is shown in Figure 10.1.

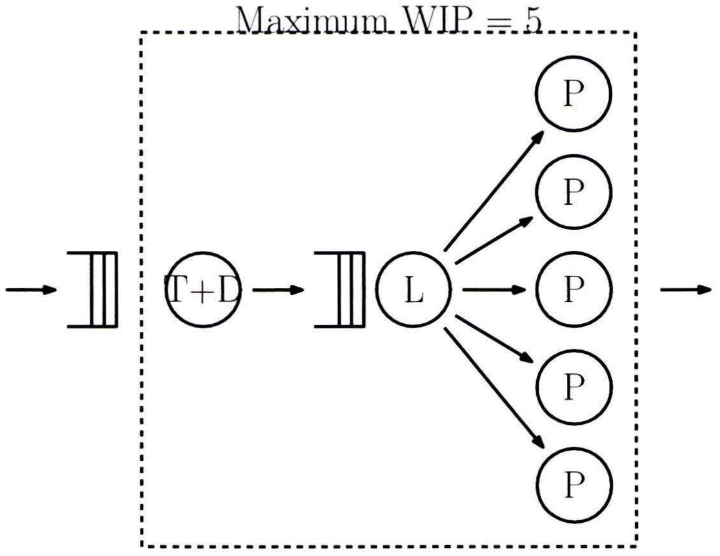


Figure 10.1: Schematic overview of the detailed model.

There is an infinite buffer to which lots arrive. From this buffer lots first go into the *Dispatching delay and transportation* part of the model. From this they travel to the *Loading* part after which they will go to the *Processing* part. The number of lots in the system is counted and a new lot can only enter the *Dispatching delay and transportation* process if there are less than five lots in the system.

The fact that the physical *Loading* process can only handle one lot at a time means that the EPTs in this section are not WIP dependent. The WIP dependency in EPT observations is caused by processing steps that are performed in parallel, as explained in Section 3.3. There are no processes that happen in parallel if there is just one lot in the system.

The EPT observations for the *Processing* part of the model are also independent of the WIP level. There can physically only be five lots on the litho cell simultaneously, and the *Processing* part is modeled as five parallel workstations. Having WIP independent EPT observations makes the distributions easier to analyse and interpret, because there is only one distribution instead of one for each WIP bucket.

The way the *Process* part is modeled also means that it is possible for lots to overtake each other in this part of the model. The odds of this happening are small because the EPT observations here have a very low variability. For our machine the coefficient of variation was approximately 0.25. Furthermore lots are only entering this section of the system after they have been through the loading part, which creates a time spacing between lots. This makes it even less likely, but not impossible, that there will be any overtaking. In the actual data there is some overtaking during this last part of the process, about half a percent of all lots overtake one or more lots here. If overtaking in the processing part of the model would be a real problem it is also possible to use a single workstation to describe it. The processing times for this workstation would then be WIP dependent, which makes it harder to analyse and change the behavior.

Unfortunately it was not possible to model the *Dispatching delay and transportation* in a way to eliminate the WIP dependency. The EPT observations for this part of the system are dependent on the WIP level in the first buffer and processing step. The circumstances that cause this can not be extracted from the data that is available. Furthermore the behavior here is very dependent on the specific customer and how their factory is managed.

The unscheduled down behavior of the system is modeled explicitly again. When an unscheduled down occurs, the entire litho cell (including *Dispatching delay and transportation*, *Loading* and *Processing*) will halt and wait until the time to repair has passed. This is similar to the earlier model, where an unscheduled down also interrupted the entire litho cell. After the repair time has passed, all parts of the litho cell will continue processing at the point where they were interrupted. This assumes that there is no scrapping of lots.

For all three parts the effective process times can be obtained by using the data in the MES file. Remember the extra information that was shown in Table 4.1. In the full aggregate model only the arrival and departure times to and from the system were used. For the new, more detailed, model this extra information is used to calculate the effective process times for each of the three parts. The EPT for *Dispatching delay and transportation* and *Loading* are calculated using the EPT algorithm for a single sever. The EPTs for the *Processing* part are calculated using the EPT algorithm for a G/G/m system [21]. The details of the EPT calculations are shown in Appendix E.

The implementation of the detailed model in SimPy can be found in Appendix C. The implementation in SimPy uses another approach than the previous two models. To keep the code more clear the SimPy internal queue manager is used. This means that it is no longer possible to model the overtaking behavior of lots. As a result only the mean cycle time can be simulated. The benefits of having a clearer and easier to maintain code outweighed the benefits of being able to simulate the entire cycle time distributions. For the simulation model it is again assumed that all the input variables are stationary and independent of each other and themselves. This model is not directly used to calculate any effects in this thesis, instead it is part of a tool set that can be used to analyze individual litho cells and predict how much improvement can be gained for certain changes to the system.

10.2 Validation

Before the new simulation model can be used it has to be validated. This validation is similar to the validation of the full aggregate model. This means that the arrivals are kept in the same order as the one found in the MES file, because there was a clear decreasing trend.

Because of this the simulation length is limited to the length of the original data again. The failure behavior is also kept in the same sequence as in the original data because randomizing them would have too much impact on the cycle times for such a short simulation time. The EPT observations for *Dispatching delay and transportation*, *Loading* and *Processing* are used as empirical distributions. One hundred independent simulations are performed and a 99% confidence interval is constructed for the mean cycle time. This confidence interval does not include the actual measured mean cycle time. Contrary to Section 5.6, the confidence interval is now quite big. The maximum is almost 10% bigger than the minimum value. The simulated mean cycle time is compared to the measured one and the deviation is defined as $\delta = \frac{\text{simulated}-\text{measured}}{\text{simulated}}$. The deviation is shown in table 10.1.

	δ
Mean	0.073752375

Table 10.1: Deviation between simulation and measured CT.

The accuracy of the detailed model is still acceptable with a deviation of 7.3% between the simulated and measured mean cycle time, although it is not as good as the full aggregate model. This can indicate that, although we did not find any, some correlations between the EPT observations for the different parts of the system are not taken into account in the detailed model. Furthermore, the spread of the 99% CI on the simulated mean cycle time is quite big. This indicates that more simulation runs are needed in order to get a reliable mean from the detailed model. However, the extra insight this model can give on changing different parts of a litho cell is worth the extra inaccuracy.

Chapter 11

Cycle Time Prediction Tool

The detailed model from Chapter 10 is the basis for a tool set to analyse litho cells at customer sites, the Cycle Time Prediction Tool (CTPT). This tooling includes automated processes to convert log files into all the required input files for the simulation model. This conversion is done in a number of steps. These steps are explained in the following sections. However, not everything is automated. The user still has to check that the inputs are stationary and independent of each other and themselves. If this is not the case the simulation model should be adjusted accordingly.

11.1 Work flow

The work flow for the CTPT is shown in Figure 11.1. There are two scripts available to process the data and convert it into inputs for the simulation model. These scripts, and the simulation model itself, come with a graphical user interfaces to control them. The first one is CLT2EPT. This is where the MES file and Status file are converted into a number of files. These files can be divided into two groups.

The first group consists of the inter-arrival times, the times to repair and the times to failure. These output files can be used in the simulation model directly. The seconds group are the EPT observations for *Dispatching delay and transportation*, *Loading* and *Processing*. These are processed by the EPT2INP script before they can be used in the simulation model. This script removes all the observations that contain an unscheduled down and sorts the observations for *Dispatching delay and transportation* into the correct WIP buckets. These EPT observations can be plotted in a similar manner to Figures 5.7, 5.8 and 5.9, to identify any outliers and their root causes.

After all the input files have been checked for trends and dependencies they can be used to run the simulation model.

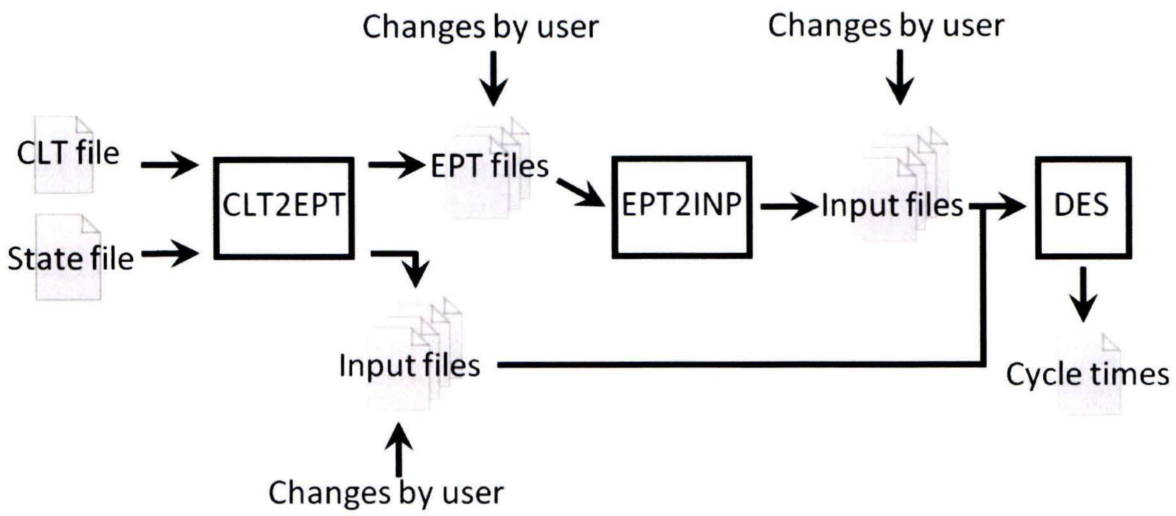


Figure 11.1: Flow of data for the CTPT.

11.2 GUI

All three scripts in the work flow come with a GUI to control them. These GUIs are explained in the following subsection.

11.2.1 CLT2EPT

The first script in the work flow is CLT2EPT. This script requires two input files, the MES file and the Status file. The GUI is shown in Figure 11.2. The location of the two input files can be selected after clicking the buttons. Pushing the start button will calculate the EPT observations (as described in Appendix E), the inter-arrival times, the times to repair and the times to failure and write them to csv files. Note that these times to repair and times to failure are based on the Status file. They can be used in the simulation later on if the user want to simulate the exact failure behavior that occurred during the data collection. This can be useful if the measured cycle times and the simulated ones are being compared. Alternatively, the times to failure and times to repair from the USD data (Figure 4.1) can be used to simulate with the global failure behavior to find the long term effects.

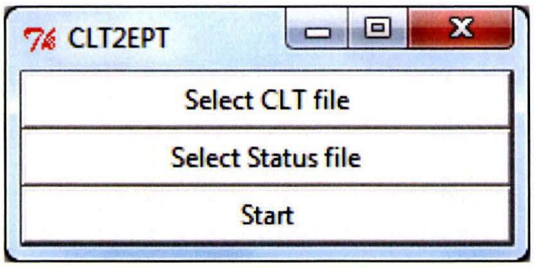
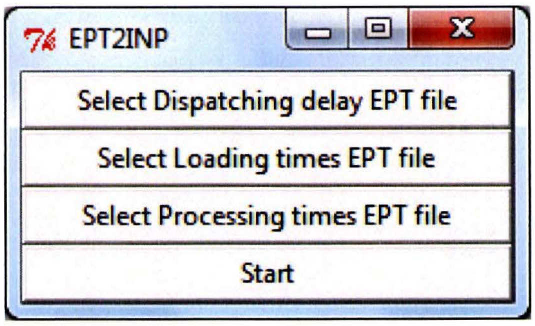


Figure 11.2: CLT2EPT GUI.


EPT2INP
Select Dispatching delay EPT file
Select Loading times EPT file
Select Processing times EPT file
Start
Figure 11.3: EPT2INP GUI.

The GUI will display a warning if the script is about to overwrite any csv files. The names of the files that will be overwritten are shown and the user can chose to abort or continue. Furthermore, there are two types of error messages that can occur: "Can not open input files!" and "Error converting CLT and Status files to EPTs!". The first message indicates that there is a problem opening the input files. The user should check if the files still exist and if another program might already be using them. The second error is a more general error, it indicates that something went wrong during the actual calculations. The user should check if the MES and Status data files are error free and in the correct format.

11.2.2 EPT2INP

The second script in the work flow is EPT2INP. It requires three input files, the csv files containing all the EPT observations for the three parts in the detailed model. The GUI is shown in Figure 11.3. The location of the these files is input using the corresponding buttons. After the start button is pressed the EPT files will be converted into input files for the simulation model. All EPTs that are observed while the machine was *Unscheduled down* are automatically removed, any other outliers should be removed prior to running this script.

The script will give a warning if files are about to be overwritten, and two types of errors. These are similar to the ones for CLT2EPT.

11.2.3 Simulation model

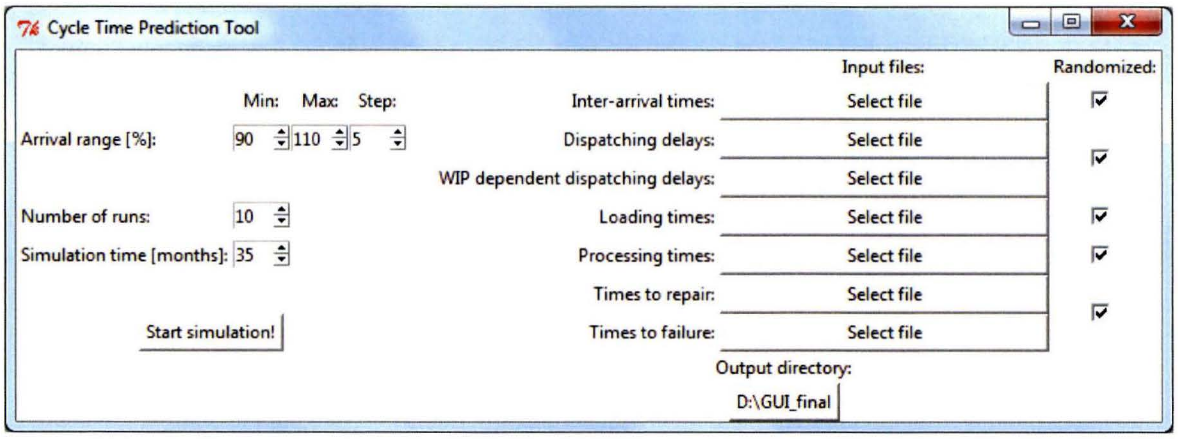
The final step of the work flow is performed by the simulation model. The GUI is shown in Figure 11.4. There are a number of simulation settings that are managed through this GUI. The first ones are the input files. Their locations are input using the buttons on the right hand side. Behind each of the input files is a check mark that indicates if the simulation model has to use the actual stream or if it should use the input file as an empirical distribution. This last option is called randomized. Simulating an actual stream can be used for inputs that contain correlations, that are not included in the model, with themselves or other inputs. Furthermore it can be used to simulate the system with the same failure behavior as during the measurements. Randomizing the failure behavior can have a big impact on cycle times and cause the predictions to deviate from the measured data.

The CTCP does not support fitted distributions, only empirical ones. *Dispatching delay and transportation* is shortened to just dispatching delay. It requires two input files, one that is sorted by WIP level and one that is ordered in chronological order. The simulation model will select the correct file from these two automatically, depending on which simulation mode (randomized or not randomized) is chosen.

The simulation length can be chosen in the bottom left spin box. The actual simulation will end as soon as this simulation length is reached, or if one of input files was ran as the actual stream and there is no more data. The arrival times can be varied to generate a Cycle time - Throughput curve. The range in which they are varied is chosen using the top left spin boxes. All the observations for the arrival times are scaled by these values, which changes the mean but leaves the coefficient of variation the same.

The number of runs at each of these inter-arrival levels is chosen using the middle spin box on the left. The simulation model will only perform one run at each level if none of the inputs are randomized, because in this case the output is deterministic.

Finally, the output directory can be chosen on the bottom right. The default location is the directory that the script is ran from. Pressing the start button will start all the simulations and a simple counter will show the progress.



Cycle Time Prediction Tool

	Min:	Max:	Step:
Arrival range [%]:	90	110	5
Number of runs:	10		
Simulation time [months]:	35		

	Input files:	Randomized:
Inter-arrival times:	Select file	✓
Dispatching delays:	Select file	✓
WIP dependent dispatching delays:	Select file	
Loading times:	Select file	✓
Processing times:	Select file	✓
Times to repair:	Select file	✓
Times to failure:	Select file	

Output directory: D:\GUI_final

Figure 11.4: GUI to run the simulation model.

Three types of error messages can occur: "Not all input files are defined!", "Not able to open output file!", "Error during simulation!". If the first error occurs the user should check if the locations for all input files have been defined correctly. The second warning indicates that the output file, called `output.csv` is in use by another program and should be closed. The final error is a more general error that means something went wrong in the actual simulation model.

If no errors were encountered a window stating "Simulation finished!" will pop up, and the output file containing the mean inter-arrival times and corresponding mean cycle times can be opened for post processing.

11.3 Test case

This test case shows the result of the Cycle Time Prediction Tool by simulating cycle times for the original distributions, and when all observations in *Processing* have been reduced by 10%. The raw output is plotted using Matlab and is shown in Figure 11.5. The 95% confidence intervals based on the same data are shown in Figure 11.6.

Many more scenarios can be thought of and simulated using this tool. A list of possible improvements and their effect can be created, which can be used to judge if an improvement is worth the time and money required to implement it.

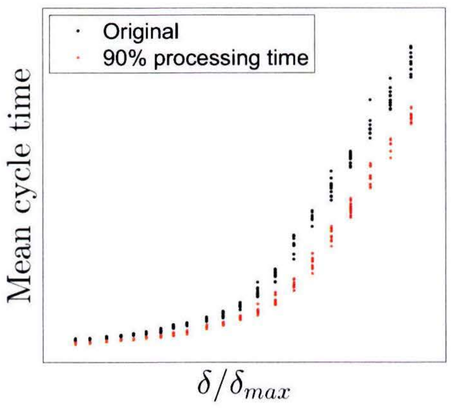


Figure 11.5: CTPT output.

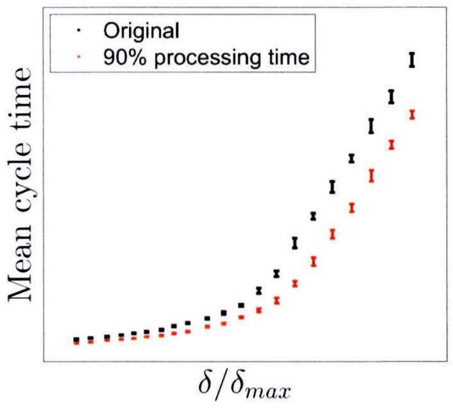


Figure 11.6: 95% CI of CTPT output.

Chapter 12

Conclusions & Recommendations

The goal of this thesis is to investigate methods to improve the cycle times for the lithography area in a semiconductor manufacturing plant. This thesis has three objectives:

1. Investigate the effect of unscheduled machine downs on the cycle times for one photolithography machine.
2. Investigate how making machines work in parallel can improve cycle times in the photolithography area.
3. Develop tooling to analyze any photolithography machine for inefficiencies and investigate how big their impact is on cycle times.

Three different models have been created so reach these objectives, one model per objective. The results for each of them are discussed in the following sections.

12.1 Unscheduled downs

To find the effect of unscheduled downs on the cycle times an aggregate simulation model of a photolithography machine is created. This model aggregates everything that happens on the photolithography machine into one processing time. A filtering method is designed to exclude the effect of unscheduled downs from these aggregated process times. The unscheduled downs that are removed this way are reintroduced by modeling the failure behavior explicitly, which makes it possible to perform a sensitivity analysis to find the effect of different breakdown behavior on the cycle time. There are five cases that are investigated, the result for each of these cases on the mean cycle time is shown in Table 12.1. These results are measured for a utilization level of approximately 90%. If the utilization of a machine is lower, the improvements on cycle time will also be lower.

These results can be used by ASML to decide if investing effort into one of these improvements is worth the reduction in cycle time.

Case	Result on mean cycle time
Reducing the mean time to repair with 10%	2.5% reduction
Reducing the COV of time to repair with 10%	1% reduction
Reducing the mean time to failure with 10%	3% reduction
Reducing the COV of time to repair with 10%	0.3% reduction
Removing the highest 10% of times to repair	12% reduction

Table 12.1: Sensitivity of cycle times to unscheduled downs.

12.2 Matching of machines

The second goal of this thesis if to investigate the effect of machine matching on cycle time. Each scanner has it's own specific imperfections which lead to specific offsets in the exposed layers. These offsets are called the machine signature or machine fingerprint. Matching machines is a method to make two machines mimic each others so called fingerprints.

In semiconductor manufacturing the relative positioning of two consecutive layers sometimes requires a precision that is on the very edge of the scanners capabilities. These layers are called critical layers. Common practice is to create these critical layers on the same machine, to avoid extra inaccuracy that another machine fingerprint will add. This practise is called machine dedication.

Machine dedication has one big disadvantage: In case of a machine failure, all dedicated WIP will have to wait until the machine is fixed before it can be processed again. This can lead to long and unpredictable cycle times. Sometimes cycle times get so out of control that dedicated lots are processed on another machine, and the resulting loss of quality is accepted.

There is an alternative way to solve this problem. It is possible to reprogram a number of machines, so they copy each others fingerprint on top of their own. This method requires a lot of time to recalibrate the machines, which has to be repeated every time an important part of one of the machines breaks and gets replaced. This makes it impractical to match more than two machines together. Of course this method is not perfect, but it can decrease the number of layers that require machine dedication.

The effect of machine matching is quantified by taking the aggregate model that was designed for the previous section, and simulate two machines in parallel instead of just one. As products arrive to these machines they are dedicated to one machine, or they can be processed on either one. The amount of products that is dedicated to one machine is varied from 0 to 100%.

Simulations with this model show that a system without any dedication has about half the cycle time of a system where all lots are dedicated. Furthermore, decreasing the amount of dedicated lots from 100 to 75% will already decrease the mean cycle time with 25%.

12.3 Cycle time prediction tooling

The last goal of this thesis is to develop a tool set that can analyse any photolithography machine, identify inefficiencies, and predict the effect on the mean cycle time when they are solved. This is done by dividing the photolithography machine into three parts, which results in a more detailed model. This makes it easier, compared to the full aggregate model, to find which part of the machine is causing inefficiencies.

The detailed model is implemented in a discrete event simulation program and a custom GUI is designed to control it. Furthermore, generating the input for the model from log files is automated and required only a few mouse clicks. This tool set makes it possible to quickly find improvements for any photolithography machine, and predict how much they can lower the mean cycle time.

12.4 Recommendations for future work

There are a number of things that can be of interest for ASML, to use the results of this thesis. These are summed up below:

- Apply the Cycle Time Prediction Tool in a real factory, find the best improvements and validate how well the predictions of the model are.
- Incorporate the Cycle Time Prediction Tool with the throughput prediction models that are already being used by ASML. Combining the impact on cycle time and throughput gives a better sense of the result of possible improvements.
- Study the effect of different unscheduled down behavior for two matched machines, instead of just one.

Furthermore, the research in this thesis has identified a number of future research topics for the EPT approach that are worth investigating:

- Investigate how EPT observations can be linked to actual lots. This makes it easier to identify outliers and their root causes.
- Investigate how the overtaking behavior can be described in a general way. This can be done by fitting a distribution, but it might also be possible to find a method to reconstruct what actually happens in a factory from arrival and departure data. This behavior can then be incorporated into the simulation model.
- In this thesis a method was designed to extract the effect of unscheduled machine downs from the EPT observations, and model it explicitly. It might be interesting to see if other effects, for instance the effect of planned maintenance, can be investigated in a similar way.

Appendix A

Aggregate simulation model in SimPy to find the effect of unscheduled downs

As per request of ASML the simulation is done in a python based discrete event simulation language called SimPy. SimPy is a process-based discrete-event simulation language based on standard Python and released under the GNU LGPL. It provides the modeler with components of a simulation model. These include processes for active components like customers, messages, vehicles or in our case machines, as well as resources for passive components that form limited capacity congestion points (like servers or checkout counters).

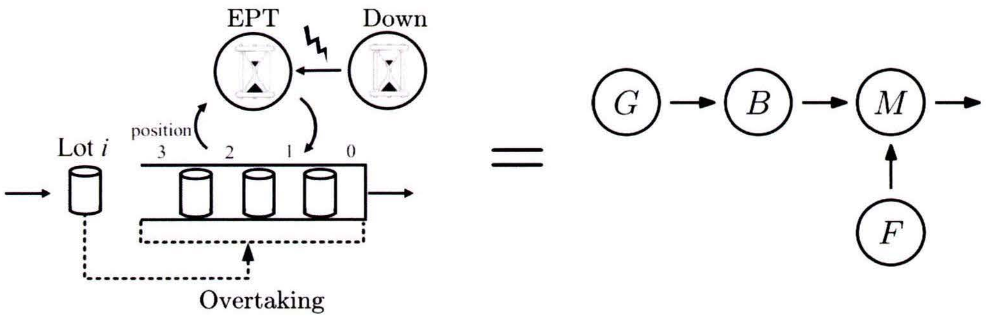


Figure A.1: EPT model and corresponding simulation model.

Figure A.1 shows the EPT model from Chapter 3, and the processes that are used for the implementation in SimPy. The SimPy implementation consists of the following processes: An arrival process G, an infinite buffer B, a machine M and a machine failure process F. The input and output for the simulation model is shown in Table A.1.

The arrival process simulates lots arriving to the system from a previous processing step. These lots arrive in buffer B where they can overtake a number of lots, depending on the overtaking behavior. The machine services products and takes the out of the buffer when it's done. The processing time of this machine is the "regular", without machine downs, WIP dependent, effective process time. The breakdown process can preemptively interrupt

Input	Output
Inter-arrival time distribution	Cycle times per lot
EPT distributions	
Overtaking distributions	
Time to repair distribution	
Time to failure distribution	

Table A.1: Input and output for the simulation model.

machine M to simulate an unscheduled down. The coding for all these processes and some general programming will be explained in detail in the following sections.

A.1 JobClass

```
class JobClass:
    def __init__(self):
        self.ArrivalTime = now()
```

The JobClass is a class that represents a lot. Upon creation of each lot the creation time is stored as an attribute. These lots will be placed in the buffer upon creation. Machine M will try to take a lot from this buffer as soon as it needs it.

A.2 Lists

Another class is created to store all the lists that the program needs.

```
class Lists:
    Idle = []
    Buffer = []
    Ta = []
    OVERT = []
    EPTS = []
    TTRS = []
    TTFS = []
```

The first one is a list of idle machines, which contains pointers to the machines that are idle. The model is programmed is such a way that a machine will only go idle if it is not processing any lot and there are no lots in the buffer to start working on, the machine is starved. Every time a new lot is generated the generator process checks if a machine is idle. If this is the case then the idle machine will be reactivated to start processing the newly arrived lot.

The second list is the Buffer. This is actually the buffer process as shown in Figure A.1. It works as a list that all the newly generated lots are inserted into. This inserting will happen according to the overtaking behavior.

The other lists are used to store the different input parameters. The inter-arrival times are stored in Lists.Ta. the overtaking distributions are stored as a list of lists in Lists.OVERT. The EPTs are stored in a similar fashion in Lists.EPTS. Finally the times to repair and times to failure are stored in Lists.TTRS and Lists.TTFS, respectively.

These lists with input parameters are used when the simulation is running the actual streams of the input parameters, or when it is using empirical distributions.

A.3 Generator

The generator is the arrival process that generates new lots, the JobClass that was discussed earlier. This process is a Simpy process, unlike the previous two classes. This means that it can interact with the build in discrete event simulation engine, adding and removing events to and from the future event list, and reactivating other SimPy processes.

```
class Arrival(Process):
   def Run(self):
      while 1:
         ta = expovariate(1.0/mu)
         yield hold, self, ta
```

The process will halt for a certain amount of time to simulate the inter-arrival time. These inter-arrival times are sampled from an exponential distribution. Once the inter arrival time has passed, a new lot is created in the form of a JobClass.

```
         J = JobClass()
         O = choice(Lists.OVERT[min(len(Lists.Buffer),35)])
```

The amount of lots that will be overtaken are sampled from an empirical overtaking distribution. This empirical overtaking distribution is WIP dependent, it is stored as a list of lists. Each of these lists holds all the values for one WIP level. The amount of overtaken lots is sampled from the corresponding WIP level. If the current WIP level is higher than the biggest WIP number that was encountered in the actual data, it will take the maximum encountered WIP level.

```
         Lists.Queue.insert(len(Lists.Buffer)-O, J)
         if Lists.Idle != []:
            reactivate(Lists.Idle[0])
```

The new lot is then inserted in the right place in the the Buffer list and a check is performed to see if there are any idle machines. The simulation is setup in such a way that any machine that is idle, is starved. A new arrival into the buffer will un-starve a machine so it can start processing again. If the list of idle machines was indeed not empty, the first machine in the list is reactivated.

A.4 Breakdown

The breakdown process is a SimPy process that simulates machine failures and will halt the Machine process. This means that the interrupted routine of the Machine process is called. This routine is explained in Section A.5.

```
class BreakDown(Process):
   TTR = None
   Failure = 0
```

There are two global variables stored in the Breakdown process. The first one is the time to repair, TTR. This is done because this variable has to be shared with the machine process because it needs to know how long it should be down. The Breakdown process also needs this value to wait until the repair time has passed before starting another time to failure delay.

The second global variable here is BreakDown.Failure. This variable is used as a flag to signal the Machine process. There are two states the machine can be in while it fails. It can be productive in which case the production delay will be interrupted to simulate the downtime. The other case is that the machine is idle. In this case the machine is reactivated and this flag is used to let the machine know that it was reactivated to simulate a machine down.

```
def Run(self, machine):
    while 1:
        ttof = gammavariate(k,t)
        yield hold, self, ttof
```

The Run routine is the main loop of the Breakdown process. The machine that it should target is given as an input argument in the routine. It will start by sampling the time to failure from a gamma distribution and continue to wait until this time has passed.

```
        BreakDown.Failure = 1
        BreakDown.TTR = gammavariate(k,t)
```

Once this time has passed global variable BreakDown.Failure is set to one and the time to repair is sampled from a gamma distribution.

```
        if Lists.Idle.count(machine) > 0:
            reactivate(machine)
        else:
            self.interrupt(machine)
        yield hold, self, BreakDown.TTR
        BreakDown.Failure = 0
```

The program then checks if the target machine is idle. If this is the case the machine is reactivated and the Breakdown.Failure flag tells the machine that it should simulate a machine down. If the machine is not idle it must be processing and it is interrupted.

The Breakdown process then delays for the time to repair, sets the global variable Breakdown.Failure to zero and starts the cycle again.

A.5 Machine

The Machine process is also a Simpy process. It simulates the aggregate machine that can sample the processing times either from a set of gamma distribution, one for each WIP level, or from the actual input as an empirical distribution.

```
class Machine(Process):
    def __init__(self,name):
        Process.__init__(self,name)
        Lists.Idle.append(self)
    def Run(self):
```

```
while 1:
    Lists.Idle.remove(self)
    if BreakDown.Failure == 1:
        yield hold, self, BreakDown.TTR
```

The main routine is again called Run, which contains an infinite loop. Once it starts the machine removes itself from the list of idle machines. It will then check if it is reactivated because it has to simulate a machine down. It does this by checking the global variable BreakDown.Failure. If this is the case the machine will simulate the required downtime.

```
    while Lists.Buffer != []:
        te = choice(Lists.EPTS[min(len(Lists.Queue)-1,7)])
```

If there was no machine failure to simulate, or the corresponding repair time has passed, the machine will check if there are any lots in the buffer to process. During normal operation at least one of these two checks should pass, because it is the only reason why the machine can be reactivated. The only exception to this is the first time the routine is called in which case neither check will pass and the machine will passivate itself, waiting for another SimPy process to reactivate it.

As long as the buffer is not empty the machine will stay in the processing loop. It will determine the processing time from a WIP dependent empirical distribution.

```
        while te > 0:
            yield hold, self, te
            if self.interrupted():
                te = self.interruptLeft
                yield hold, self, BreakDown.TTR
                self.interruptReset()
            else:
                break
```

While the processing time is bigger than 0 it will stay in the processing loop. The machine will delay until either the processing time has passed, or until the Breakdown process interrupts this delay. If it was interrupted `te` is updated to the remaining processing time and the machine failure is simulated. After this is done the interrupt routine is ended and the while loop for the processing time will repeat itself. Multiple failures can occur this way while processing the same lot.

This way of modeling assumes that no lots are scrapped during a machine failure. This makes it a bit easier to model instead of having another decision weather or not to scrap the lot. This approach seems justified because the impact of the rework on cycle time is expected to be low compared to the impact of the actual machine failure.

```
        J = Lists.Buffer.pop(0)
        print("%f" % (now() - J.ArrivalTime))
    Lists.Idle.append(self)
    yield passivate, self
```

Finally, after the machine has simulated the processing time the first lot is taken from the buffer and the difference between the current time and the arrival time of the lot is printed. If the stocker is empty it will break from the while loop and place itself on the idle machine list and passivate itself, waiting for another process to re-activate it.

A.6 Initialization and main loop

First the SimPy simulation engine is initialized using the initialize() command. After this all the inter-arrival times, times to failure and times to repair are imported from their respective comma separated value (csv) files.

```
def main():
    initialize()
    ta = csv.reader(open('ta_excl_failures.csv', 'rb'), delimiter=',')
    for line in ta:
        Lists.Ta.append(int(line[0]))
    ttf = csv.reader(open('ttf.csv', 'rb'), delimiter=',')
    for line in ttf:
        Lists.TTFS.append(int(line[0]))
    ttr = csv.reader(open('ttr.csv', 'rb'), delimiter=',')
    for line in ttr:
        Lists.TTRS.append(int(line[0]))
```

Next the EPTs are imported from the EPT csv file. In this file all EPTs that correspond to one WIP level are on the same row. The same is done for the overtaking data. Note that each row has a different number of values. In the csv format all rows must have the same length however, because of this every row that has less entries than the longest row is filled with empty strings until all rows have the same length. This means that for every entry in these csv files it has to be checked if it is an empty string or an actual value.

```
    te = csv.reader(open('epts.csv', 'rb'), delimiter=',')
    n = 0
    for line in te:
        Lists.EPTS.append([])
        for i in range(len(line)):
            if len(line[i].strip()) > 0:
                Lists.EPTS[n].append(int(line[i]))
        n += 1

    overts = csv.reader(open('overt.csv', 'rb'), delimiter=',')
    n = 0
    for line in overts:
        Lists.OVERT.append([])
        for i in range(len(line)):
            if len(line[i].strip()) > 0:
                Lists.OVERT[n].append(int(line[i]))
        n += 1
```

Finally all different SimPy processes that were discussed above are initialized and started, and the simulation is started. The maximum simulation time is passed to the program from the command line.

```
    M = Machine(name="M1")
    activate(M,M.Run())
    A = Arrival()
    activate(A,A.Run())
    B = BreakDown()
    activate(B,B.Run(machine = M, arrivals = A))
    MaxSimtime = float(sys.argv[1])
    simulate(until=MaxSimtime)
```

Appendix B

Implementation of the matching model in SimPy

The model to simulate the matching of two machines is also implemented in SimPy. This model is an extended version of the model that is shown in Appendix A. The differences are presented in this appendix.

B.1 Two machines

The first difference is that instead of one machine, there are two. These machines have the same processing and failure behavior, but sample independent of each other. This is schematically shown in figure B.1.

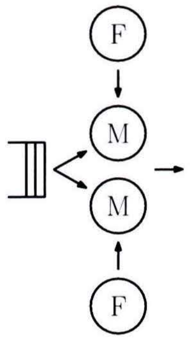


Figure B.1: Aggregate simulation model for matching.

B.2 Arrival process

The arrival process is the same as the one in Chapter A, except that there is now a new choice and an extra list. After the amount of overtaken lots is determined, there is a choice whether or not this lot is dedicated to one machine. `random()` returns a uniform distributed value on the interval [0.0, 1.0). This value is compared to the percentage of dedication that is being simulated, `pdedicated`. If the random value is smaller, another distribution is sampled to determine which machine the lot is dedicated too, `D`. If it is bigger the lot is dedicated to machine 0, which in this case means it can be processed on both machines. The job is then added to the queue. `D` is added to another queue. This queue has the same length as the original queue and `D` is inserted in the same position as the job.

```
class Arrival(Process):
   def Run(self):
      while 1:
         ta = expovariate(1.0/mu)
         yield hold, self, ta
         J = JobClass()
         if random() < pdedicated:
            D = int(round(uniform(1,2)))
         else:
            D = 0
         Lists.Queue.append(J)
         Lists.QueueM.append(D)
         for m in Lists.Idle:
            reactivate(m)
```

B.3 Machine process

In Appendix A the machine process only checked if there was a lot available in the buffer. If this was the case the machine started producing. This procedure is now extended. The machine will check the queue for lots that can be processed on either machine and for lots that can only be processed by the specific machine itself. It will count the number of each of these lots and store it respectively to LQ0 and LQD.

```
class Machine(Process):
   produced = 0
   def __init__(self,name,number):
      Process.__init__(self,name)
      self.number = number
      Lists.Idle.append(self)
   def Run(self):
      while 1:
         Lists.Idle.remove(self)
         if BreakDown.Failure == 1:
            yield hold, self, BreakDown.TTR
         LQ0 = Lists.QueueM.count(0)
         LQD = Lists.QueueM.count(self.number)
```

The machine will keep processing as long as there are lots in the queue that can be processed on the current machine. It will find the first lot in the queue that it can process. This lot is then removed from the queue to make sure it is not process by two machines at the same time. The process time is taken from the empirical WIP dependent EPT distribution. The amount of WIP is decided by counting the number of products in the queue that is dedicated to the machine, plus half the number of lots in the queue that can be processed by either machine.

```
while LQ0 + LQD > 0:
   if LQ0 > 0 and LQD > 0:
      type0 = Lists.QueueM.index(0)
      typeD = Lists.QueueM.index(self.number)
      J = Lists.Queue.pop(min(type0,typeD))
      D = Lists.QueueM.pop(min(type0,typeD))
   elif LQ0 > 0:
      type0 = Lists.QueueM.index(0)
      J = Lists.Queue.pop(type0)
      D = Lists.QueueM.pop(type0)
   elif LQD > 0:
      typeD = Lists.QueueM.index(self.number)
      J = Lists.Queue.pop(typeD)
      D = Lists.QueueM.pop(typeD)
   te = choice(Lists.EPTS[min(ceil(LQ0/2)+LQD-1,7)])
   while te > 0:
      yield hold, self, te
      if self.interrupted():
         te = self.interruptLeft
         yield hold, self, BreakDown.TTR
         self.interruptReset()
      else:
         break
   Machine.produced += 1
Lists.Idle.append(self)
yield passivate,self
```

Appendix C

Implementation of the detailed model in SimPy

The detailed model in Figure 10.1 is implemented in the SimPy language to create a discrete event simulation model. For the detailed model we are only interested in the average cycle time. This means that the overtaking behavior is not taken into account anymore and it is no longer required to manage the queue explicitly. Instead, the built in SimPy queueing engine can be used. This means that the modeling concept is different. Instead of the machine and queue, the lots and their path through the litho cell are now modeled. The components of this model are explained in this Appendix.

C.1 makePorts

The makePorts process creates five ports for the litho cell. These ports can be claimed by lots. The SimPy queueing engine will create and maintain a FIFO queue for lots that try to claim a port while there are none available.

```
class makePorts(Process):
   def run(self,port):
      for i in range(nports):
         yield put,self,port,["Port%02d"%(i,)]
```

C.2 Generator

The generator is very similar to the generator from Appendix A. The only difference is that this time a different process is activated. A Lot process is activated this time, which simulated the entire production process. This is contrary to the earlier model where a JobClass was activated and added to the queue. This queue was then clear by the Machine process.

```
class Source(Process):
   def generate(self,port,dd,load):
      while 1:
            ta = choice(Lists.ta)
            yield hold,self,ta
            l = Lot(name = "Lot%02d"%(i,))
            activate(l,l.process(port=port,dd=dd,load=load))
            i += 1
```

C.3 BreakDown

The BreakDown process simulates the unscheduled down behavior of a machine. In the earlier model there was one machine that would be interrupted. In this modeling approach all the lots that are being processed on the litho cell have to be identified and interrupted.

```
class BreakDown(Process):
   def generate(self,load,ddr,port):
      while 1:
         ttf = choice(Lists.ttf)
         BreakDown.TTR = choice(Lists.ttr)
         yield hold,self,ttf
         free_ports = nports-len(Lists.onTrack)
```

The time to failure and time to repair is sampled from an empirical distribution. As soon as the time to failure has passed the amount of free ports is determined. `Lists.onTrack` is used to keep track of which lots are being processed on the litho cell. The amount of free ports is equal to the total number of ports, minus the amount of lots that are being processed.

```
         for j in list(set(Lists.onTrack)-set(load.waitQ)-set(ddr.waitQ)):
            self.interrupt(j)
         if free_ports > 0:
            yield get,self,port,free_ports
            claimdports = self.got
```

The lots that are being processed on the machine are interrupted, unless they are not in a delay state. This is the case if they are waiting for the dispatching delay process, or the loading process. The free ports on the machine are then claimed so no new lot can arrive and be processed, while a machine outage is simulated.

```
         yield hold, self, BreakDown.TTR
         if free_ports > 0:
            yield put,self,port,claimdports
```

The downtime is then simulated, and afterwards the ports that were claimed are released again.

C.4 Litho cell

The main process of a lot going through all production steps is coded from the lot's perspective. A lot will try to claim a port on the track as soon as it is created by the generator process. If no port is available it will queue until it gets one. The queueing handled by the internal queueing engine. As soon as a port is available for the lot it will add itself to the list of lots that are being processed.

```
class Lot(Process):
   def __init__(self,name):
      Process.__init__(self,name)
      self.ArrivalTime = now()
   def process(self,port,dd,load):
      yield get,self,port,1
      whichPort=self.got[0]
      Lists.onTrack.append(self)
```

The lot then waits to claim capacity on the dispatching delay process, simulates the dispatching delay, and releases the dispatching delay process.

```
      yield request, self, dd
      t_delay = choice(Lists.ddwd[min(len(port.getQ)+len(Lists.onTrack)-1,
         len(Lists.ddwd)-1)])
      yield hold, self, t_delay
      while t_delay > 0:
         yield hold, self, t_delay
         if self.interrupted():
            t_delay = self.interruptLeft
            yield hold, self, BreakDown.TTR
            self.interruptReset()
         else:
            break
      yield release, self, dd
```

The same is done to simulate the loading process.

```
      yield request, self, load
      t_load = choice(Lists.load)
      yield hold, self, t_load
      while t_load > 0:
         yield hold, self, t_load
         if self.interrupted():
            t_load = self.interruptLeft
            yield hold, self, BreakDown.TTR
            self.interruptReset()
         else:
            break
      yield release, self, load
```

Processing is always accessible so there is never any queueing necessary.

```
      t_processing = choice(Lists.processing)
      while t_load > 0:
         yield hold, self, t_processing
         if self.interrupted():
            t_processing = self.interruptLeft
            yield hold, self, BreakDown.TTR
            self.interruptReset()
         else:
            break
      yield put,self,port,[whichPort]
      Lists.onTrack.remove(self)

      print now() - self.ArrivalTime
```

C.5 Initialization

The initialization is similar to the one explained in Appendix A. The only difference is that in this model a number of resources are initialized, instead of just the processes. There are three types of resources: The number of ports, the dispatching delay process and the loading process.

```
port = Store(name='FOUP ports', unitName='ports', capacity=5,
    initialBuffered=None, putQType=FIFO, getQType=FIFO, monitored=False,
    monitorType=Monitor)
dd = Resource(name="Dispatching delay", capacity=1)
load = Resource(name="Load delay", capacity=1)

p = makePorts()
activate(p,p.run(port=port))
s = Source(name='Source')
activate(s,s.generate(port=port, dd=dd, load=load))
if len(Lists.ttr) > 0 and len(Lists.ttf) > 0:
    f = MachineFailure(name='MachineFailure')
    activate(f,f.generate(load=load, dd=dd, port=port))
simulate(until=simtime)
```

Appendix D

Algorithm to calculate EPTs and overtaking behavior

The algorithm to calculate the EPT and overtaking observations is based on the algorithm proposed by Veeger [21]. The original algorithm is shown here.

```
for row in MESData:
    i = row[2]
    ev = row[1]
    t = row[0]
    if ev == "A":
        if len(xs) == 0:
            s = t
            sw = 1
        xs.append([i, len(xs)])
    elif ev == "D":
        EPT.writerow([t-s, sw])
        ys = []
        while len(xs) > 0:
            j = xs[0][0]
            aw = xs[0][1]
            xs.remove(xs[0])
            if j < i:
                ys.append([j, aw])
            elif j == i:
                xs = ys + xs
                k = len(ys)
                break
        Overt.writerow([k, aw])
        if len(xs) > 0:
            s = t
            sw = len(xs)
```

The following variables are used in this algorithm: variable t denotes the event time, variable ev the event type (either an arrival a or a departure d), and i the lot arrival number (so lot i is the i^{th} arriving lot). Furthermore, variable xs is a list that stores for each lot in the system its arrival number, i, and the number of lots in the system just before its arrival aw. Variable s is used to store the EPT start time. Variable sw stores the number of lots in the system just after the EPT start. Variable k denotes the number of lots that a lot has overtaken. The overtaking behavior is calculated using a list ys that stores part of list xs. Variable j stores a lot arrival number.

The EPT algorithm takes the aggregate model viewpoint. Upon an arrival event, a new EPT is started if the lot arrives in an empty system (`len(xs) = 0`). The start time s becomes t and the corresponding WIP level is stored in variable sw. For every arriving lot, the lot arrival number i and the number of lots in the system just before arrival (`len(xs)`) are added to the end of list xs. When a departure event occurs, an EPT ends, the EPT being current time t minus EPT start time s. The EPT is written to output along with number of lots in the system just after the EPT start sw.

Next, the algorithm reconstructs how many lots k were overtaken by the departing lot by iteratively removing each lot from xs and assigning its arrival number and the number of lots just before its arrival to variables j and aw respectively. If the arrival number of the observed lot is lower than the arrival number i of the departed lot, then this lot was overtaken and the information of the overtaken lot is added to ys. The departing lot is found in list xs if the arrival number j of the observed lot is equal to i. The list of lots still in the system is created by merging lists ys and xs. The length of ys is equal to the number of lots that arrived earlier than lot i, but that are still in the system upon the departure of lot i. In other words, the length of ys is equal to the number of lots overtaken by lot i. This is written to the output, along with the number of lots aw in the system just before arrival of the lot.

If there are still lots in the system after the departure (`len(xs)>0`), a new EPT start time is stored in s, as well as the corresponding number of lots currently in the system (`len(xs)`).

This original algorithm is slightly adjusted to combine the EPT observations with the machine status information. To do this an extra check is included into the code before the EPT is written to the output. This check compares the starting and ending times of the EPT observation with the starting and ending times of the machine states from the Status file. The beginning en ending times of the unscheduled downs are stored in list $Dtimes$. The beginning and ending times of all scheduled downs are stored in list $SDtimes$.

If an EPT observation has any overlap with an unscheduled down, it is marked as "EPT during unscheduled down". If an EPT observation has any overlap with a scheduled down, but not with an unscheduled down, it is marked as an "EPT during scheduled down". Overlap occurs when an EPT start takes place during a machine down, an EPT end takes place during a machine down, or a machine down takes place between an EPT start and end. In short: An EPT observation is marked as EPT during a down if the following is true:

$$\begin{aligned}(Outage_{start} \le EPT_{start} \le Outage_{end}) \text{ or } (Outage_{start} \le EPT_{end} \le Outage_{end}) \\ \text{or } (EPT_{start} < Outage_{start} \text{ and } EPT_{end} > Outage_{end})\end{aligned} \tag{D.1}$$

Line 11 in the algorithm is replaced with the following code to implement this functionality:

```
for usd in Dtimes:
    if (s > usd[0] and s < usd[1]) or (t > usd[0] and t < usd[1]) or (s < usd[0]
        and t > usd[1]):
            EPT.writerow([t-s, sw, "USD"])
            break
    else:
        for sd in SDtimes:
            if (s > sd[0] and s < sd[1]) or (t > sd[0] and t < sd[1]) or (s < sd[0]
                and t > sd[1]):
                EPT.writerow([t-s, sw, "SD"])
                break
        else:
            EPT.writerow([t-s, sw, "U"])
```

Appendix E

Calculating effective process times for the detailed model

The effective process times for all three production steps are calculated from the data in the MES file. These EPTs are not calculated as straightforwardly as in chapter 5, because of the additional restrictions in the model. Additional data is needed to calculate the EPTs for all production steps. Instead of just the arrival and departure times to the litho cell we now need the arrival times, the times the lot arrive on the port of the litho cell, the times the wafers of a lot are loaded into the track and the departure times from the litho cell. These times are abbreviated to A, T, L and D.

E.1 Dispatching delay and transportation

The time between a port becoming free on and a lot actually arriving on the track is called the dispatching delay. This time includes anything that happens in between these two events, for instance transportation from the stocker to the track, inspection of the lot by an operator, waiting for WIP constraints to resolve, and so on. This process is modeled as a single server and the algorithm used to calculate the EPT observations is shown here.

```
for row in OverallData:
    t = row[0]
    ev = row[1]
    i = row[2]
    if ev == "A":
        inq += 1
        if inq == 1 and ontrack < nports:
            s = t
            sw = 1
```

The first part of the algorithm is the same as the one described in chapter 5: OverallData is a chronological list with the times different events happened, and which lot was involved. First the time, type of event and lot id are stored. If the event is an arrival to the stocker `inq` is increased by one. This variable is used to keep track of the amount of lots that are waiting in the stocker, to be transferred onto the track.

An EPT observation is started if this is the first lot to arrive in the stocker, and the amount of lots on the track is smaller than the number of ports.

```
elif ev == "T":
    inq -= 1
    ontrack += 1

    for usd in Dtimes:
        if (s_d > usd[0] and s_d < usd[1]) or (t > usd[0] and t < usd[1]) or (
            s_d < usd[0] and t > usd[1]):
            ddf.writerow([t-s, sw, "Unscheduled Down"])
            break
    else:
        for usd in SDtimes:
            if (s_d > usd[0] and s_d < usd[1]) or (t > usd[0] and t < usd[1])
                or (s_d < usd[0] and t > usd[1]):
                ddf.writerow([t-s, sw, "Scheduled Down"])
                break
        else:
            ddf.writerow([t-s, sw, "Up"])

    if inq > 0 and ontrack < nports:
        s = t
        sw = inq
```

If the event is a lot arriving on the track the amount of lots in the queue is reduced by one. Then the EPT is written to the dispatching delay csv file along with the corresponding machine state and starting WIP level. A new EPT observations is started if there is at least one lot in the queue, and the number of lots on the track is smaller than the number of ports.

```
elif ev == "D":
    ontrack -= 1
    if inq > 0 and ontrack == nports - 1:
        s = t
        sw = inq
```

The final part of the EPT calculation algorithm is including the departure event, when a lot leaves the track after it has been processed. As soon as this happens one of the ports on the track is free again, so a new lot can be mounted on it. If this is the case and there is at least one lot waiting in the stocker, a new EPT observation is started.

E.2 Loading

The loading EPTs describe everything that happens between the time a lot has arrived to the track, and the time all the wafers in this lot have been loaded into the track. This consists for the most part of the opening the FOUP and the loading the wafers into the track, by the wafer handler robot.

```
elif ev == "T":
    onload += 1
    if onload == 1:
        s = t
```

inq is used to keep track of the number on the loading part of the system. If there are not lots in this system, a new EPT observation is started as soon as a new lot arrives. Note that we do not keep track of the starting WIP here. This is because in the loading process there are no significant processing steps that can happen in parallel, making this EPT distribution independent of the WIP.

```
elif ev == "L":
    for usd in Dtimes:
        if (s > usd[0] and s < usd[1]) or (t > usd[0] and t < usd[1]) or (s <
            usd[0] and t > usd[1]):
            loadf.writerow([t-s, "Unscheduled Down"])
            break
    else:
        for usd in SDtimes:
            if (s > usd[0] and s < usd[1]) or (t > usd[0] and t < usd[1]) or (s
                < usd[0] and t > usd[1]):
                loadf.writerow([t-s, "Scheduled Down"])
                break
        else:
            loadf.writerow([t-s, "Up"])
    onload -= 1
    if onload > 0:
        s = t
```

The EPT observation is ended as soon as all wafers for a lot have been loaded into the track. It is then written to the loading csv file, along with the corresponding machine state. If there is at least one lot in the in the sytem a new EPT observation is started.

E.3 Processing

The processing times describe the rest of the processing on the litho cell. This includes for instance applying the photosensitive layer, exposing, post processing, loading the wafers back into the FOUP and waiting to be transportation from the port to the next processing step. The processing of lots is modeled as five parallel machines. The EPTs are calculated based on the EPT algorithm for the m-server aggregate model [21].

```
elif ev == "L":
    inprocessing += 1
    rs.append([i,t])
```

As soon as a lot has been loaded into the track it has arrived in the processing section of the model. An EPT observation is started if the number of lots in processing is smaller than the number of ports.

The original algorithm includes a section in case there are more lots in the system than the number of servers. In our case this is not possible. There can never be more lots in the litho cell than the number of ports, so this part of the algorithm is removed. If there are errors in the log file that is used this might not hold. The variable inprocessing is used to keep track of the number of lots in the processing section. This variable can be check if is any suspicion of a faulty log file.

```
elif ev == "D":
    inprocessing -= 1
    d = rs.pop([x[0] for x in rs].index(i))
    s = d[1]

    for usd in Dtimes:
        if (s > usd[0] and s < usd[1]) or (t > usd[0] and t < usd[1]) or (s <
            usd[0] and t > usd[1]):
            tailf.writerow([t-s, "Unscheduled Down"])
            break
    else:
        for usd in SDtimes:
            if (s > usd[0] and s < usd[1]) or (t > usd[0] and t < usd[1]) or (s
                < usd[0] and t > usd[1]):
                tailf.writerow([t-s, "Scheduled Down"])
                break
        else:
            tailf.writerow([t-s, "Up"])
```

As soon as a lot leaves the litho cell the ETP observation for the processing section is written to the corresponding csv file. Because the processing steps

Appendix F

Unscheduled down correction algorithm

```
i = 0
while i < len(failures) - 1:
    t_start = failures[i][0]
    t_end = failures[i][1]
    t_start_next = failures[i+1][0]
    ttf = t_start_next - t_end
    b = 0
    while ttf < ttf_min:
        b = b + 1
        t_end = failures[i+b][1]
        t_start_next = failures[i+1+b][0]
        ttf = t_start_next - t_end
    i = i + b + 1
    print t_start , t_end
```

In this algorithm `failures` is a list of lists. Every list in here contains the starting time and the ending time for each unscheduled down. Each machine has its own list of lists, and each of these lists is sorted on the starting times. We start to calculate the time to failure between the first and the second failure. If this time is smaller than the average production time, `ttf_min`, the time to failure between the second and third failure is calculated and compared to `ttf_min` and so on. Once a time to failure bigger or equal to `ttf_min` is found all previous failures are concatenated into one failure. This process is repeated until all entries in `failures` are checked.

Note that this script will produce an out of index error if the last and second to last failure in `failures` are not separated by `ttf_min`. This is not a problem because if this is the case we can not be certain that this last machine down has indeed finished. If we concatenate the last two failures in this case it might only be part of the actual machine down time.

Bibliography

[1] Interview with ASML startup engineer, April 2012.

[2] S. Aarts. *Influence of Outages on LithoCell Performance.* 2003.

[3] I. Adan and J. Resing. *Queueing Theory: Ivo Adan and Jacques Resing.* Eindhoven University of Technology. Department of Mathematics and Computing Science, 2001.

[4] D. Babbs and R. Gaskins. Effect of reduced equipment downtime variability on cycle time in a conventional 300mm fab. In *Advanced Semiconductor Manufacturing Conference, 2008. ASMC 2008. IEEE/SEMI*, pages 237–242, 2008.

[5] FabTime Editors. Conquering wip bubbles. *FabTime Cycle Time Management Newsletter*, 8(5):4–8, June 2007.

[6] W. Hopp and M. Spearman. *Factory Physics Second Edition.* McGraw-Hill/Irwin, 2000.

[7] F. J. A. Jansen. Ept based aggregate modeling for an mri department: the case of limited number of arrival and departure events. 2012.

[8] J. F. C. Kingman. On queues in heavy traffic. *Journal of the Royal Statistical Society. Series B (Methodological)*, 24(2), 1962.

[9] A. Kock. *Effective Process Times for Aggregate Modeling of Manufacturing Systems.* 2008.

[10] C. Kuo, C. Chien, and J. Chen. Manufacturing intelligence to exploit the value of production and tool data to reduce cycle time. *Automation Science and Engineering, IEEE Transactions on*, 8(1):103 –111, jan. 2011.

[11] M. Lentz. Industry economic model & enterprise value of cycle time. Presented at the SEMATECH Symposium Taiwan, Hsinchu, Taiwan, September 2011.

[12] A. Malventano. An Inside Look at Intel and Micron 25nm Flash Memory Production. `http://www.pcper.com/reviews/Storage/Inside-Look-Intel-and-Micron-25nm-Flash-Memory-Production/Inside-Fab`, Feb. 2010. Accessed: 14/03/2013.

[13] F. J. Massey. The Kolmogorov-Smirnov test for goodness of fit. *Journal of the American Statistical Association*, 46(253):68–78, 1951.

[14] M. E. Pfund, S. J. Mason, and J. W. Fowler. *Handbook of Production Scheduling*, chapter 9 : Semiconductor Manufacturing Scheduling and Dispatching, pages 213–241. Springer, 2006.

[15] A. Schoemig. On the corrupting influence of variability in semiconductor manufacturing. In *Simulation Conference Proceedings, 1999 Winter*, volume 1, pages 837–842 vol.1.

[16] SEMATECH. Semiconductor Manufacturing Process. `http://www.sematech.org/corporate/news/mfgproc/mfgproc.htm`. Accessed: 14/03/2013.

[17] J. Shin, S. Lee, J. Yeo, H. Kim, J. Lee, and W. Han. Study of machine to machine overlay error for sub-60-nm memory devices. *Journal of Vacuum Science & Technology*, 26(6), 2008.

[18] G. D. Taylor and S. Heragu. A comparison of mean reduction versus variance reduction in processing times in flow shops. *International Journal of Production Research*, 37(9):1919–1934, 1999.

[19] J. van der Eerden. Litho area productivity improvement. 2004.

[20] J. van der Eerden, T. Saenger, W. Walbrick, H. Niesing, and R. Schuurhuis. Litho area cycle time reduction in an advanced 300mm semiconductor manufacturing line. In *Advanced Semiconductor Manufacturing Conference, 2006. ASMC 2006. The 17th Annual SEMI/IEEE*, pages 114–119, 2006.

[21] C. Veeger. *Aggregate modeling in semiconductor manufacturing using effective process times*. 2010.

[22] J. Vervoort and J. Rooda. *Learning Timed χ*. 2007.